Couplage Planification et Ordonnancement: Approche hiérarchique et décomposition

Olivier Guyon

Université d'Angers **Année 2010, N° ordre 1037**

Couplage Planification et Ordonnancement : Approche hiérarchique et décomposition

Thèse de doctorat

Spécialité Informatique

École doctorale Sciences et Technologies de l'Information et de Mathématiques

Présentée et soutenue publiquement

le 19 mai 2010

à Angers

par **Olivier Guyon**

Devant le jury ci-dessous :

Présidente du jury :	Pr. N. Brauner, Université Joseph Fourier de Grenoble
Rapporteurs :	Pr. P. Chrétienne, Université Pierre & Marie Curie (Paris VI) Pr. S. Dauzère-Pérès, École des Mines de Saint-Étienne
Examinateur :	Dr. V. Barichard, Université d'Angers
Directeur de thèse :	Pr. É. Pinson, Institut de Mathématiques Appliquées d'Angers
Co-encadrants :	Dr. P. Lemaire, Institut National Polytechnique de Grenoble Dr. D. Rivreau, Institut de Mathématiques Appliquées d'Angers

Laboratoire d'Ingénierie des Systèmes Automatisés - Angers

Institut de Mathématiques Appliquées d'Angers

École des Mines de Nantes

ED N° 503

Remerciements

Je tiens à remercier en tout premier lieu mon *Grand Chef* : Monsieur Éric Pinson -Professeur à l'Institut de Mathématiques Appliquées d'Angers-. Sa confiance, sa disponibilité sans faille, ses remarques et conseils toujours avisés, sa patience dans ses explications mais aussi ses *mises sous pression* bien-fondées ont toujours été axés vers un unique but : ma progression. Bien plus qu'un directeur de thèse, il a su -dans un climat de travail idéal- m'insuffler une partie de sa passion débordante pour la Recherche Opérationnelle et m'a soutenu à chaque moment clé de mon début de carrière. Je lui serai donc, pour ces raisons et bien d'autres encore (notamment le raccourcissement de la peine promise de 17 ans de doctorat), à jamais reconnaissant ; je suis même particulièrement fier de désormais appartenir à sa *filiation* scientifique. J'exprime par ailleurs tout aussi vivement ma gratitude à Monsieur David Rivreau -Directeur et Maître de Conférences à l'Institut de Mathématiques Appliquées d'Angers- et Monsieur Pierre Lemaire -Maître de Conférences à l'Institut National Polytechnique de Grenoble- pour m'avoir co-encadré. Merci à David pour son implication réelle au quotidien. Ses indications, remarques, critiques et idées fécondes se sont toutes révélées d'un intérêt majeur. Il a parfaitement su gérer le thésard parfois difficile que j'ai été. Toute ma gratitude va également à Pierre. Malgré des conditions géographiques qui auraient pu compliquer sa tâche, je l'ai toujours senti disponible et investi. Ses réflexions sensées et sa minutie ont joué un rôle prépondérant dans ma (notre) réussite. Sans fausse hypocrisie, je pense sincèrement avoir eu des conditions d'encadrement idéales que je souhaite à tout doctorant.

J'adresse tous mes remerciements à Monsieur Philippe Chrétienne -Professeur à l'Université Pierre & Marie Curie- et Monsieur Stéphane Dauzère-Pérès -Professeur à l'École des Mines de Saint-Étienne- qui m'ont fait l'honneur d'être rapporteurs de ces travaux. Je remercie également Madame Nadia Brauner -Professeur à l'Université Joseph Fourier de Grenoble- et Monsieur Vincent Barichard -Maître de Conférences à l'Université d'Angers- pour leur participation, en tant qu'examinateurs, à ce jury.

Parce que ma vie de thésard sans eux aurait été bien terne, je veux ici exprimer toute mon amitié à mes deux collègues de la salle de Recherche : Célinette la plus chouette et Bobo le héros. La chaleur humaine, le soutien et aussi les conneries que chacun d'eux apportait au bureau ont été un vrai support à mon bien-être. Merci donc à eux deux pour toutes ces séances de lancer de pingouins avec/et sans tête, d'air guitar au sol, de battage de record du monde d'asseyage sans chaise contre un mur, de délires sur le hit interplanétaire *Week-end*, de démonstration de judo, de cours de couture autour de portefeuille, ... et encore plein d'autres inoubliables instants que la clause de *Secret Défense* m'oblige à taire ici.

Je n'oublie pas non plus tous mes autres collègues de l'Institut de Mathématiques Appliquées et de l'École des Mines de Nantes. Que ce soit à Nantes ou à Angers, j'ai été parfaitement accueilli et intégré. Je veux remercier plus particulièrement

Christine qui, malgré les *p'tit con* amicaux dont elle avait l'habitude de me qualifier (souvent à juste titre je dois maintenant l'avouer), s'est toujours montrée avenante et prête à m'aider. Pour exactement les mêmes raisons (mais sans le sobriquet qui va avec cette fois-ci), un chaleureux merci à Anne (qui connait désormais la date de mon anniversaire par cœur) et Antony (qui, en tant que voisin de bureau, a aussi su tolérer l'ambiance bruyante et parfois peu travailleuse qu'instauraient mes deux comparses de la Salle de Recherche). Je remercie aussi très sincèrement Christelle et Jacky. Sans leur tractation dans l'ombre pour combattre une bizarrerie contractuelle, le travail présenté dans ce manuscrit n'aurait très probablement jamais abouti.

Je remercie aussi tous mes amis qui m'ont accompagné durant ces trois années et quelques de thèse (et même depuis beaucoup plus longtemps pour la plupart). Ils m'ont apporté toutes les joies, délires, rires, sorties en tout genre tellement nécessaires à mon épanouissement extra-professionnel. Merci donc à tous ces généreux amis : Béjamin, Burel, Chaban, Du Will, Elo Zélaye, Fifou, Fophie, La mouche, Gégé le coup de boule, Gui la since, Jean-Phi marée, Juju, Jujute, La carpe, La Gwen, Ma(o)rie, Mon p'tit Vinvin, Rélie, Séb Pin(ot)eau, Super Nanou, La tenaille, Virguinie.

Je veux terminer cette partie si importante du manuscrit en remerciant les personnes qui me sont le plus chères, à savoir l'ensemble de ma famille : Maman, Papa, Mamie, Pépé, Céline, Jérôme, Sébastien, Cholé et Tathan. Sans leur soutien inconditionnel, je n'aurais très probablement pas pu aller au bout de ce projet. Un merci plus particulier tourné envers mes deux parents ; je vous suis et serai à jamais reconnaissant de votre aide sans faille dans chacun de mes choix personnels et professionnels (même les plus difficiles). Votre confiance totale en chacune de mes décisions et le soutien solide qui s'ensuivait m'ont toujours donné une assurance indispensable.

Table des matières

Introduction générale

Dans un contexte industriel où la mondialisation exacerbe une concurrence désormais internationale, *accroissement de productivité* rime plus que jamais avec *augmentation de profits* et *gain de parts de marché*.

La planification de la production et la gestion des ressources humaines constituent incontestablement deux piliers organisationnels influant de manière directe sur ce facteur-clé qu'est la productivité. D'un côté, la planification de la production cherche à affecter des ressources (humaines ou matérielles) à différentes tâches ou missions à réaliser [LRKB77,CC88,Leu04,Pin04,BN04]. D'un autre côté, la gestion de personnel tend, la plupart du temps, à réduire les coûts de main d'œuvre [BM90, Sch99, MS03, EJKS04, SPR05].

Les acteurs du monde économique ont bien compris les enjeux conséquents qui se cachaient derrière ces deux postes stratégiques. De nombreuses ressources (humaines et financières) sont en pratique souvent affectées à la gestion de ces deux ressorts de l'entreprise. Cependant, la complexité sans cesse croissante des règles (physiques, législatives et sociales) auxquelles est soumise chaque société rend le travail des décideurs de plus en plus complexe.

Devant un tel constat, il apparaît alors clair que la Recherche Opérationnelle, qui *propose des modèles conceptuels pour analyser des situations complexes et permet aux décideurs de faire les choix les plus efficaces*[1], peut s'avérer d'une très grande utilité ; d'autant plus que les forces de calcul que les nouvelles technologies sont désormais en mesure de nous offrir ont sans nul doute permis d'ouvrir de nombreuses pistes d'exploration à ce domaine de la Recherche encore à l'aube de son histoire.

Dans le monde du transport, de nombreuses avancées scientifiques, prenant en compte simultanément ces deux aspects organisationnels que sont la gestion des ressources humaines et la planification de la production (ici des trajets), ont permis de réaliser des économies substantielles. En ce sens, on peut par exemple mentionner certains travaux traitant de telles problématiques intégrées [BGAB83,DDI$^+$94,FH96, CSSD01, KJN$^+$02, FHW03, MCS05].

En comparaison, le domaine manufacturier semble avoir été étrangement laissé-pour-compte. Peu de travaux font en effet état d'une gestion intégrée de ces deux fonctions de l'entreprise. Le problème est alors souvent résolu, en pratique, dans un processus de décision à deux niveaux clairement sous-optimal : on décide dans un pre-

1. Présentation de la R.O. par JC Billaut : http ://www.roadef.org/content/road/pdf/ROetGI.pdf

mier temps des ressources humaines à employer et on gère ensuite l'organisation de la production. Quelques publications annonciatrices -et aux résultats encourageants- d'une prise en considération de ce problème réel à enjeux considérables sont tout de même à citer. Artigues et al. proposent par exemple dans [AGR07, AGRV09] un état de l'art sur ces problématiques (dans les domaines de la production et du transport), mais aussi une formalisation et une technique de résolution hybride alliant Programmation Linéaire et Programmation Par Contraintes. Dans ses travaux, Hooker [Hoo05,Hoo07] s'intéresse lui aussi à cette problématique intégrée et propose également une technique de résolution alliant Programmation Linéaire et Programmation Par Contraintes. L'approche qu'il définit repose sur une décomposition de type Benders [Ben62], l'un des sous-problèmes induits étant résolu par des techniques classiques de la Programmation Linéaire, et l'autre, par celles de la Programmation Par Contraintes. D'autres techniques de résolution ont aussi été implémentées. Bailey et al. [BAL95] ainsi que Alfares et Bailey [AB97] étudient une heuristique fondée sur les concepts de la programmation dynamique. Danniels et Mazzola [DM94] proposent, pour leur part, une méthode arborescente exacte ainsi qu'une heuristique décomposant le problème en trois niveaux (chaque niveau étant résolu indépendamment) puis itérant sur la recherche de solutions réalisables améliorantes.

Comme l'indique son titre, cette thèse traite des problèmes intégrant planification d'agents et ordonnancement de production. Le principal objet des trois années de recherche qu'a constitué le travail présenté dans ce mémoire a consisté à éprouver des techniques de décomposition et génération de coupes dédiées pour ce genre de problématiques ; une telle décomposition ayant déjà été appliquée avec succès par Detienne et al. [Det07, DPPR09] pour un problème de planification d'agents avec une charge de travail prédéterminée. Nous visions donc initialement à étendre le cadre d'application de ce genre de techniques (a priori prometteuses), afin de valider -ou invalider- leur intérêt sur cette classe de problèmes à décomposition naturelle intuitive.

La première problématique intégrant planification d'agents et ordonnancement de production à laquelle nous nous sommes intéressés durant cette thèse consiste, sommairement, à affecter au moindre coût des horaires de présence à des opérateurs diversemment qualifiés ; la quantité et la qualité des ressources humaines ainsi induites devant permettre de définir, pour l'outil de production, l'ordonnancement complet d'un ensemble de jobs préemptifs symbolisant les demandes des clients.

Après avoir défini précisément la problématique, nous en formulons deux modélisations sous forme de Programmes Linéaires en Nombres Entiers ; la première découlant naturellement du problème, la seconde -plus compacte- exploitant l'aggrégation de certaines données et variables de décision.

Ce seconde modèle est alors exploité afin de définir deux bornes inférieures par décomposition lagrangienne ; la première dualisant l'ensemble des contraintes cou-

plantes, et la seconde ne s'intéressant qu'à la dualisation des contraintes couplant l'attribution des compétences aux opérateurs à l'affectation de leurs horaires de présence. Ces approches s'avèrent malheureusement toutes deux peu compétitives au regard de la borne inférieure obtenue, via un solveur de programmation linéaire, par la relaxation continue du problème. La première décomposition présente l'inconvénient, du fait de la dualisation de l'ensemble des contraintes couplantes, de laisser une part trop importante de la complexité inhérente au problème à la méthode de sous-gradient ; cette dernière peinant alors à converger. La seconde décomposition, pour sa part, palie en partie à cela. Cependant, l'un des deux sous-problèmes lagrangiens qu'elle induit (*flot maximal à coût minimum*) se montre particulièrement consommateur en temps CPU ; le temps de calcul de la fonction duale en chaque itération devenant alors rédhibitoire et rend cette seconde approche par décomposition lagrangienne peu adaptée.

Deux méthodes de résolution exacte par décomposition et génération de coupes sont ensuite présentées. La première est basée sur une décomposition classique d'un programme linéaire mixte (décomposition de Benders [Ben62]) dont nous détaillons le développement théorique. La seconde permet de résoudre le problème par une technique analogue de décomposition et génération de coupes, les coupes étant cette fois-ci spécifiquement dédiées à la structure de la formalisation compacte. Nous montrons comment on peut, relativement à une affectation d'horaires de présence aux opérateurs, trouver une éventuelle contrainte de production non respectée par la recherche d'un flot maximal dans un graphe orienté décomposé en niveaux. Le processus de résolution se traduit alors par les résolutions successives de programmes maîtres (*sac à dos multi-choix multidimensionnel*, problème *NP-difficile* [MT87, MT90]), de contrôles de réalisabilité et d'ajout de contraintes. Deux variantes sont proposées pour résoudre les programmes maîtres : une résolution exacte par un solveur de programmation linéaire et une résolution heuristique par l'application de l'heuristique de Feasibility Pump [FGL05, BFL05, AB07]. Si l'approche par décomposition de Benders s'avère peu compétitive face à une résolution directe par un solveur de programmes linéaires, l'autre approche se révèle particulièrement efficace ; elle permet en effet la résolution exacte -et heuristique- d'instances générées aléatoirement en un temps réduit.

Afin de parfaire notre étude, nous proposons aussi une technique de génération d'inégalités valides applicable en pré-process de toute méthode de résolution. Cette dernière s'appuye sur les concepts du *raisonnement énergétique* [LEE92, BLPN99]. Elle permet de générer très rapidement un ensemble de coupes complémentaires aux inégalités de réalisabilité des techniques de décomposition décrites auparavant. Les résultats expérimentaux démontrent de manière très probante que l'usage de ces coupes initiales dites *énergétiques* améliore très significativement les performances de chaque approche présentée dans ce document.

Forts des résultats encourageants de la technique dédiée de décomposition et

génération de coupes obtenus lors des premiers travaux de la thèse, nous avons entrepris d'expérimenter cette même approche sur une autre problématique intégrant, elle aussi, planification d'agents et ordonnancement de production. Cette nouvelle étude fait l'objet de la seconde partie de ce manuscrit.

Ces travaux furent l'occasion de nous consacrer à une problématique où l'ordonnancement de production ne se calculait plus en un temps polynomial. Nous considérons en effet dans cette seconde étude un problème de production *NP-complet* : la version décisionnelle d'un job-shop. Dans un problème classique de job-shop, central en Théorie de l'Ordonnancement, n jobs doivent s'exécuter sur m machines selon une séquence opératoire propre à chacun ; chaque machine ne pouvant traiter qu'un job à la fois, et les jobs ne pouvant être exécutés sur plusieurs machines simultanément. Le critère d'optimisation attaché à un tel problème varie selon les études ; on peut par exemple citer le critère retenu dans nos travaux : la minimisation de la durée totale d'ordonnancement.

Cette nouvelle étude nous donna également l'opportunité de tester nos techniques sur des instances connues de la littérature. La problématique à laquelle nous nous intéressons coïncide avec un des problèmes étudiés par Artigues et al [AGRV09].

Après avoir formulé cette seconde problématique, nous en présentons une formalisation sous forme d'un Programme Linéaire en Nombres Entiers. Nous nous intéressons après cela à un problème au cœur de chacune des méthodes de résolution décrites par la suite : le problème de job-shop à période fixées d'inactivité. Ce cas particulier de job-shop est tout d'abord défini, deux modèles en termes de programmes linéaires sont ensuite établis. Le premier découle intuitivement de la nature du problème, et le second exploite la réécriture de chaque période d'inactivité en un job fictif. Cette seconde formalisation s'apparente alors à celle d'un job-shop classique, pour lequel la littérature déborde de techniques de résolution performantes [CP89, CPPR04, Pin04, BN04, BFS08].

Trois catégories d'inégalités valides, générées en pré-process de toute technique de résolution, sont ensuite détaillées. La plus singulière d'entre elles concerne les coupes dites de *probing*. Elle présente l'attrait d'exploiter les mêmes fondements théoriques que les techniques de *probing* utiles à la résolution des Programmes Linéaires en Nombres Entiers [Sav94]. Dans cette idée, des variables de décision sont successivement fixées à certaines valeurs de leur domaine de définition, et des conséquences logiques impactant les autres variables du modèle sont alors recherchées. À l'instar de la première étude, de telles inégalités initiales s'avèrent particulièrement efficaces en pratique et ce, quelque soit la technique de résolution utilisée par la suite.

Une première méthode exacte, analogue à la technique de décomposition et génération de coupes mise en place pour résoudre la première problématique considérée durant cette thèse, est détaillée après cela. Bien qu'exploitant les mêmes concepts,

la technique de coupes se révèle malheureusement peu efficace ici. À l'inverse de la première problématique, la qualité des coupes de réalisabilité générées ne permet pas d'invalider un grand nombre de solutions. Cumulé avec une forte dégénérescence du problème de planification d'agents, la méthode de coupes converge alors très difficilement, allant même jusqu'à quasiment s'apparenter à une technique énumérative basique.

Non contents de cet échec, nous avons persisté et développé une méthode exacte alternative. Cette seconde technique repose à la fois sur une approche arborescente de type Procédure de Séparation et Évaluation Séquentielle et sur la méthode de coupes définie auparavant. Cette méthode hybride présente l'avantage d'utiliser les atouts de la technique de coupes (même décomposition du problème en deux sous-problèmes et ajout des mêmes coupes de réalisabilité en cours de processus) sans en subir ses inconvénients ; elle permet en effet de mieux réguler la dégénérescence du problème de planification d'agents et ne requiert pas nécessairement la résolution optimale coûteuse des programmes maîtres. Testée sur les instances d'Artigues et al. [AGRV09], cette approche s'avère particulièrement adaptée à notre problématique ; ses résultats dominent en effet ceux obtenus avec l'un des meilleurs solveurs commerciaux actuels (Ilog Cplex 11.2).

Première partie

Planification d'emplois du temps et ordonnancement de production

Introduction

Dans cette première partie, nous nous intéressons à un cas particulier de problème intégrant planification d'agents et ordonnancement de production. Nous nous plaçons pour cela dans un contexte manufacturier où le décideur doit gérer au moindre coût les emplois du temps d'opérateurs diversement qualifiés, de façon à fournir à son outil de production une quantité et une qualité de ressources humaines permettant l'ordonnancement complet d'un ensemble de jobs symbolisant les demandes de ses clients.

D'un côté, nous devons résoudre un cas particulier de problème de gestion d'emplois du temps consistant à affecter au moindre coût des séquences d'horaires de présence à des opérateurs différemment qualifiés. Par ailleurs, nous devons ordonnancer un ensemble de jobs préemptifs sur leur domaine d'exécution respectif. Ces deux problématiques d'entreprise sont corrélées dans cette étude par le fait que chaque job requiert, à tout moment d'exécution, un opérateur maîtrisant une certaine compétence. Il convient alors de fournir au décideur la distribution de coût minimal d'emplois du temps aux opérateurs de telle manière que cette dernière définisse un ensemble de ressources permettant d'ordonnancer l'ensemble des jobs de manière exhaustive.

Après avoir précisé la formulation de cette problématique particulière couplant planification d'agents et ordonnancement de production, nous en proposons deux modélisations sous forme de Programme Linéaire en Nombres Entiers. Nous exploitons alors l'une d'elles pour définir deux bornes inférieures obtenues au travers de décompositions lagrangiennes. Deux méthodes exactes sont ensuite proposées, l'une basée sur une décomposition de Benders [Ben62] et l'autre sur une technique spécifique de décomposition et de génération de coupes. Une technique de génération d'inégalités valides pouvant être appliquée en pré-processing à chacune des méthodes de résolution décrites auparavant est détaillée après cela. Nous discutons enfin de la performance de chaque technique au travers de leurs résultats obtenus sur des instances générées.

1 Formulation

Le problème étudié ici consiste en l'ordonnancement d'un ensemble de jobs (tâches) indépendants au moyen d'un ensemble de ressources (opérateurs), chaque job devant être exécuté par un opérateur maîtrisant une compétence spécifique.

Les niveaux de décision en jeu dans cette problématique sont l'affectation de cycles de travail aux opérateurs et l'ordonnancement de l'ensemble des jobs. La configuration des cycles de travail et les jobs à exécuter représentent les données en entrée de notre problème.

1.1 Roulements

Les cycles de travail sont établis dans notre étude au travers un ensemble Ω de *roulements*. Un roulement $\omega \in \Omega$ définit un canevas d'horaires de présence ou d'absence sur l'horizon de planification. Typiquement, un roulement peut représenter une logique manufacturière de travail en trois-huit, ou à jour de repos différé, ... Cette formalisation permet donc la prise en compte de nombreuses contraintes législatives, contractuelles ou autres (congés posés...).

1.2 Opérateurs et Compétences

Chaque instance fait intervenir un ensemble O de m opérateurs ainsi qu'un ensemble C de compétences. Pour chaque opérateur $o \in O$, on dispose de l'ensemble $C_o \subseteq C$ des compétences qu'il maîtrise et de l'ensemble $\Omega_o \subseteq \Omega$ des roulements qui peuvent lui être affectés. Chaque couple roulement ω - opérateur o est caractérisé par un coût η_ω^o symbolisant le coût de main d'œuvre engendré lorsque l'opérateur o se voit affecter le roulement ω. Ce coût peut dépendre des horaires de travail mais aussi du statut de l'opérateur.

1.3 Jobs

Un ensemble J de n jobs indépendants est défini pour chaque instance. Chaque job j possède une durée p_j, un domaine d'exécution $D_j = [\![r_j, d_j]\!]$, et requiert pour son exécution un opérateur maîtrisant une compétence $c_j \in C$. Les jobs sont préemptifs et peuvent, sous réserve de contraintes de compétence et de disponibilité, être exécutés par différents opérateurs. On ne peut cependant effectuer plus d'une unité d'un même job par instant. Il ne peut donc y avoir qu'au plus un opérateur affecté à un job sur un instant t.

1.4 Objectif

L'objectif du problème est de déterminer un ordonnancement de l'ensemble des jobs en affectant un roulement à chaque opérateur, de telle sorte que chacun des besoins en main d'œuvre (effectif et compétence) soit rempli au moindre coût.

1.5 Structure du problème

Il y a ici deux niveaux de décision. Il est tout d'abord nécessaire d'affecter un roulement à chaque opérateur. Ensuite, à chaque instant où l'opérateur est considéré comme présent selon le roulement qui lui a été attribué, nous devons décider quelle compétence il utilise concrètement de manière à pouvoir déterminer un ordonnancement complet des jobs.

1.6 Exemple d'instance

À travers cette étude, nous considérons un exemple d'instance à 3 opérateurs (o_1, o_2, o_3), 2 compétences (c_1, c_2), 3 jobs (j_1, j_2, j_3) et 2 roulements contenant chacun une unique période d'activité ($\omega_1 = [\![0, 8[\![$, $\omega_2 = [\![8, 16[\![$).

o_1 et o_2 maîtrisent la compétence c_1 et o_3 maîtrise c_1 et c_2. Les autres caractéristiques de cette instance (que nous noterons par la suite Exemple 1) sont décrites dans le tableau 1.

	j_1	j_2	j_3
r_j	0	2	8
d_j	10	8	16
p_j	9	2	3
c_j	c_1	c_1	c_2

	o_1	o_2	o_3
$\eta^o_{\omega_1}$	10	7	2
$\eta^o_{\omega_2}$	5	3	9

TABLE 1 – Caractéristiques de Exemple 1

Une solution réalisable (coût : 26) pour Exemple 1 consiste à affecter ω_1 à o_1 et o_2, et ω_2 à o_3. Le diagramme de Gantt (figure 1) illustre alors un plan de production possible pour cette solution.

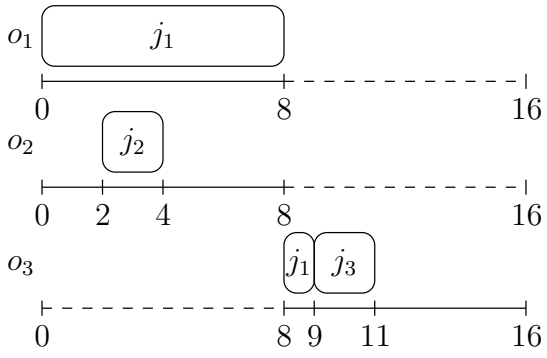

FIGURE 1 – Diagramme de Gantt

Chapitre 1

Formalisations

Dans ce premier chapitre, nous présentons deux formalisations du problème. Une modélisation indexée sur le temps découlant intuitivement de la formulation de la problématique est tout d'abord proposée. Certaines variables de décision de cette première modélisation sont ensuite agrégées afin de proposer un second modèle plus compact. Cette seconde formalisation constitue d'ailleurs le modèle de base de chaque méthode de résolution décrite dans ce document.

1.1 Formalisation indexée sur le temps

Nous proposons ici une première modélisation du problème sous forme d'un Programme Linéaire en Nombres Entiers. Cette première formalisation intuitive est indexée sur le temps.

1.1.1 Données

Les données en entrée de notre problème sont :
- H représente l'horizon de planification
- J est l'ensemble des jobs
- c_j est la compétence requise par le job $j \in J$
- D_j décrit la fenêtre temporelle d'exécution du job $j \in J$
- O est l'ensemble des opérateurs
- $C = \bigcup_{o \in O} C_o$ est l'ensemble des compétences, C_o représentant l'ensemble des compétences maîtrisées par l'opérateur o
- $\Omega = \bigcup_{o \in O} \Omega_o$ est l'ensemble des roulements, Ω_o représentant l'ensemble des roulements qui peuvent être affectés à l'opérateur o
- σ_ω est le vecteur caractéristique sur l'horizon du roulement ω : $\sigma_\omega^t = 1$ si le roulement ω est travaillé à l'instant t, $\sigma_\omega^t = 0$ sinon
- η_ω^o est le coût d'utilisation du roulement ω par l'opérateur o

1.1.2 Variables de décision

Les variables de décision que nous définissons pour cette formalisation sont :
- $x_{jt} = 1$ si une unité du job j est exécutée à l'instant t, 0 sinon
- $y^o_\omega = 1$ si le roulement ω est affecté à l'opérateur o, 0 sinon
- $z_{oct} = 1$ si l'opérateur o utilise la compétence c à l'instant t, 0 sinon

1.1.3 Programme Linéaire

Une première formalisation $[Q]$, indexée sur le temps, de notre problématique s'écrit alors :

$$[Q] : \min \quad \sum_{o \in O} \sum_{\omega \in \Omega_o} \eta^o_\omega \cdot y^o_\omega \tag{1.1}$$

$$\forall o \in O \quad \sum_{\omega \in \Omega_o} y^o_\omega = 1 \tag{1.2}$$

$$\forall j \in J \quad \sum_{t \in D_j} x_{jt} = p_j \tag{1.3}$$

$$\forall t \in H, \forall c \in C \quad \sum_{\substack{j \in J| \\ c_j = c}} x_{jt} - \sum_{\substack{o \in O| \\ c \in C_o}} z_{oct} = 0 \tag{1.4}$$

$$\forall t \in H, \forall o \in O \quad \sum_{c \in C_o} z_{oct} - \sum_{\omega \in \Omega_o} \sigma^t_\omega \cdot y^o_\omega \leq 0 \tag{1.5}$$

$$\forall o \in O, \forall \omega \in \Omega_o \quad y^o_\omega \in \{0, 1\} \tag{1.6}$$

$$\forall j \in J, \forall t \in H \quad x_{jt} \in \{0, 1\} \tag{1.7}$$

$$\forall o \in O, \forall c \in C_o, \forall t \in H \quad z_{oct} \in \{0, 1\} \tag{1.8}$$

Les contraintes (1.2) astreignent l'affectation d'exactement un roulement à chaque opérateur.

L'exécution complète des jobs durant leur fenêtre d'exécution est assurée par (1.3).

Pour leur part, les contraintes (1.4) indiquent que, à chaque instant t, on effectue exactement autant d'unités de jobs requérant la compétence c qu'il y a d'opérateurs utilisant c sur t.

Enfin, les contraintes (1.5) forcent les opérateurs à n'utiliser qu'au plus une compétence sur chaque instant où ils sont présents (selon le roulement qui leur est attribué) et exactement 0 lorsqu'ils sont absents.

1.2 Formalisation compacte

Afin de réduire le nombre de variables du modèle indexé sur le temps présenté ci-dessus, nous proposons ici une seconde formalisation basée sur une représentation par intervalles de temps et sur une agrégation des opérateurs en profils de compétences.

1.2.1 Représentation par intervalles de temps

Nous réduisons tout d'abord les instants considérés dans le modèle en ne nous intéressant plus qu'à des intervalles de temps choisis. Pour cela, nous extrayons de H les instants *significatifs*, à savoir la date de disponibilité r_j et la date échue d_j de chaque job $j \in J$, ainsi que les bornes des intervalles de présence de chaque roulement $\omega \in \Omega$. Les instants ainsi *extraits* sont ensuite triés par ordre croissant puis appariés par paires successives de manière à former k_{max} intervalles de temps I_k ($k \in K = [\![1, k_{max}]\!]$) disjoints deux à deux ; par définition $H = \cup_{k \in K} I_k$ et $\forall k \neq k' : I_k \cap I_{k'} = \emptyset$. Une couverture complète de H est ainsi déterminée.

Notons que chaque instant t non *extrait* est dit *inactif* car aucun job ne peut débuter ni ne doit se terminer, et aucun opérateur ne peut commencer ni ne doit arrêter de travailler en t. Ainsi, sur chaque intervalle I_k, les instants sont équivalents entre eux en termes de coût et de disponibilité des ressources (opérateurs et compétences).

Pour Exemple 1, comme l'indique le schéma 1.1, 4 intervalles de temps peuvent être définis : $I_1 = [\![0, 2[\![$, $I_2 = [\![2, 8[\![$, $I_3 = [\![8, 10[\![$ et $I_4 = [\![10, 16[\![$.

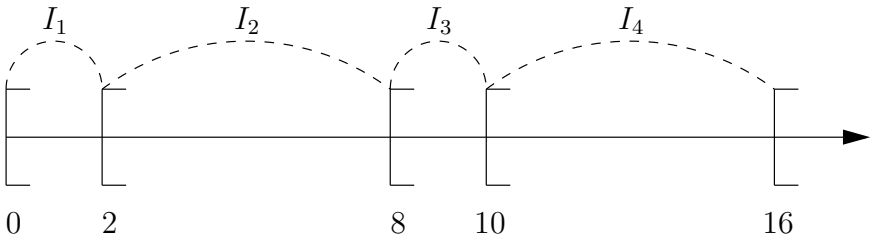

FIGURE 1.1 – Intervalles de temps pour Exemple 1

1.2.2 Agrégation des opérateurs en profils de compétences

Nous agrégeons ensuite les variables dites de *ressources opérateurs-compétences* (variables z_{oct}) en profils de compétences. Ces variables n'agissent pas directement dans la fonction coût du problème. Il est donc inutile de différencier les opérateurs possédant exactement les mêmes compétences lors de l'affectation des compétences aux instants. L'ensemble $\Theta \subseteq \mathcal{P}(C)$ est alors défini pour désigner l'ensemble des profils de compétences $\theta \in \Theta$.

Chaque opérateur $o \in O$ se voit affecter un unique profil de compétences θ_o. Deux opérateurs distincts partagent le même profil de compétences θ si et seulement s'ils maîtrisent tous les deux uniquement et exactement chaque compétence de θ.

Un profil de compétences peut alors être vu comme une généralisation des opérateurs au sens de leurs compétences.

O	C_o	Θ_o
o_1	$\{c_1\}$	θ_1
o_2	$\{c_1\}$	θ_1
o_3	$\{c_1, c_2\}$	θ_2

TABLE 1.1 – Profils de compétence pour Exemple 1

Pour Exemple 1, deux profils de compétences peuvent être définis, $\theta_1 = \{c_1\}$ pour o_1 et o_2 et $\theta_2 = \{c_1, c_2\}$ pour o_3. Le tableau 1.1 schématise l'attribution de ces deux profils de compétence θ_1 et θ_2.

1.2.3 Variables de décision

Nous définissons ici de nouvelles variables de décision pour cette autre formalisation :

- x_{jk} est le nombre d'unités du job j exécutées durant l'intervalle de temps I_k
- $y^o_\omega = 1$ si le roulement ω est affecté à l'opérateur o, 0 sinon
- $z_{\theta ck}$ est le nombre d'unités de compétence c utilisées par le profil de compétences θ durant I_k

1.2.4 Programme Linéaire

Une seconde formalisation $[P]$, dérivant de $[Q]$, peut alors être proposée :

$$[P] : \min \quad \sum_{o \in O} \sum_{\omega \in \Omega_o} \eta^o_\omega \cdot y^o_\omega \tag{1.9}$$

$$\forall o \in O \quad \sum_{\omega \in \Omega_o} y^o_\omega = 1 \tag{1.10}$$

$$\forall j \in J \quad \sum_{\substack{k \in K | \\ I_k \subseteq D_j}} x_{jk} = p_j \tag{1.11}$$

$$\forall k \in K, \forall c \in C \quad \sum_{\substack{j \in J | \\ c_j = c}} x_{jk} - \sum_{\substack{\theta \in \Theta | \\ c \in \theta}} z_{\theta ck} = 0 \tag{1.12}$$

$$\forall k \in K, \forall \theta \in \Theta \quad \sum_{c \in \theta} z_{\theta ck} - \sum_{\substack{o \in O | \\ \theta_o = \theta}} \sum_{\substack{\omega \in \Omega_o | \\ I_k \subseteq \omega}} l_k \cdot y^o_\omega \leq 0 \tag{1.13}$$

$$\forall o \in O, \forall \omega \in \Omega_o \quad y^o_\omega \in \{0, 1\} \tag{1.14}$$

$$\forall j \in J, \forall k \in K \quad x_{jk} \in [\![0, \min(p_j, l_k)]\!] \tag{1.15}$$

$$\forall \theta \in \Theta, \forall c \in \theta, \forall k \in K \quad z_{\theta ck} \in [\![0, l_k \cdot \text{card}\,\{o \in O|_{\theta_o = \theta}\}]\!] \tag{1.16}$$

où l_k désigne la longueur de l'intervalle I_k, les autres données étant analogues à celles utilisées pour la formalisation $[Q]$ indexée sur le temps.

Les techniques de résolution proposées dans cette étude sont basées sur cette seconde formalisation car elle s'avère, dans la pratique, nettement plus efficace en termes de temps de calcul.

1.2.5 Reconstitution d'une solution complète indexée sur le temps

Dans cette section, nous expliquons comment, depuis une solution optimale (notons-la $(\bar{x}^*, \bar{y}^*, \bar{z}^*)$) du modèle agrégé $[P]$, il est toujours possible de reconstituer de manière polynomiale une solution optimale du modèle $[Q]$ indexé sur le temps.

$\bar{y}^*$ définit, pour chaque opérateur, un roulement d'affectation. Ce même groupe de variables est impliqué dans les deux formalisations $[P]$ et $[Q]$. Ce sont de plus les seules variables influant sur la fonction-objectif commune aux deux modèles $[P]$ et $[Q]$. $\bar{y}^*$ est alors par conséquent, de par l'équivalence des formalisations, également solution optimale de $[Q]$.

Par la suite, on notera $\bar{\omega}_o^*$ le roulement $\omega \in \Omega$ affecté à l'opérateur $o \in O$ selon $\bar{y}^*$, c'est-à-dire : le roulement $\omega \in \Omega / \bar{y}_\omega^{o*} = 1$. Le respect de la contrainte (1.10) implique que $\bar{\omega}_o^*$ est défini et nécessairement unique pour chaque opérateur.

Les variables $\bar{x}_{jk}^*$ $(j \in J, k \in K)$ définissent, pour leur part, le nombre d'unités du job j à réaliser durant l'intervalle de temps I_k. $\bar{x}^*$ est une solution réalisable de $[P]$, il existe alors au moins une distribution $\bar{z}^*$ en ressources (ici les compétences des profils de compétences) permettant l'exécution, sur chaque intervalle de temps, de la production définie par $\bar{x}^*$.

On sait de plus que, par construction, les effectifs en opérateurs (et donc en compétences) sont constants durant chaque intervalle de temps I_k.

Dès lors, à tout plan de production (indexé sur les instants) vérifiant les niveaux de production induits par les variables $\bar{x}^*$ sur chaque intervalle de temps I_k, correspond nécessairement une affectation réalisable en ressources opérateurs-compétences.

Par conséquent, pour trouver une solution complète dérivant de $(\bar{x}^*, \bar{y}^*, \bar{z}^*)$, il suffit de déterminer une solution indexée sur le temps vérifiant les niveaux de production induits par $\bar{x}^*$ durant chaque intervalle de temps I_k.

Trouver une solution complète indexée sur le temps revient alors à fixer une distribution (existant nécessairement) des variables de décision x et z du modèle $[Q]$, répondant aux exigences du problème d'existence $[SIT]$ [2] suivant :

2. *SIT* étant acronyme de Solution Indexée sur le Temps

$[SIT]$: Existe-t-il une distribution des variables x et z vérifiant :

$$\forall j \in J, \forall k \in K \qquad \sum_{t \in D_j \cap I_k} x_{jt} = \bar{x}^*_{jk} \tag{1.17}$$

$$\forall t \in H, \forall c \in C \qquad \sum_{\substack{j \in J| \\ c_j = c}} x_{jt} - \sum_{\substack{o \in O| \\ c \in C_o}} z_{oct} \cdot \sigma^t_{\bar{\omega}^*_o} = 0 \tag{1.18}$$

$$\forall t \in H, \forall o \in O \qquad \sum_{c \in C_o} z_{oct} \leq 1 \tag{1.19}$$

$$\forall j \in J, \forall t \in H \qquad x_{jt} \in \{0, 1\} \tag{1.20}$$

$$\forall o \in O, \forall c \in C_o, \forall t \in H \qquad z_{oct} \in \{0, 1\} \tag{1.21}$$

(1.17) assure que pour chaque job et durant chaque intervalle de temps, le niveau de production induit par la solution $\bar{x}^*$ de $[P]$ est exactement respecté.

Pour sa part, (1.18) indique que, à chaque instant t, on effectue exactement autant d'unités de jobs requérant la compétence c qu'il y a d'opérateurs utilisant c à t.

(1.19) force enfin les opérateurs à utiliser au plus une compétence à chaque instant.

Les contraintes de $[SIT]$ sont l'expression des contraintes d'un problème de *flot maximum* sur un graphe orienté décomposable en niveaux ; notons-le $G_{SIT} = (X, U)$, avec

- Ensemble des sommets : $X = \{\alpha\} \cup JK \cup HC \cup OH \cup \{\beta\}$
  - α : source
  - JK : produit cartésien de J et K
  - HC : produit cartésien de H et C
  - OH : produit cartésien de O et H
  - β : puits
- Ensemble des arcs : $\{a, b\} \in U$ - (capacité γ_{ab}) -
  - $\forall a \in JK : (\alpha, a) \in U$, avec $\gamma_{\alpha a} = \bar{x}^*_{j_a k_a}$
  - $\forall a \in JK, \forall b \in HC : (a, b) \in U \Leftrightarrow (c_{j_a} = c_b) \wedge (t_b \in D_{j_a} \cap I_{k_a})$, avec $\gamma_{ab} = 1$
  - $\forall a \in HC, \forall b \in OH : (a, b) \in U \Leftrightarrow (t_a = t_b) \wedge (c_a \in C_{o_b}) \wedge \left(\sigma^{t_b}_{\bar{\omega}^*_{o_b}} = 1\right)$, avec $\gamma_{ab} = 1$
  - $\forall a \in OH : (a, \beta) \in U$, avec $\gamma_{a\beta} = 1$

La figure 1.2 donne l'allure du graphe G_{SIT} dans le cas général.

Dans la mesure où on a déjà établi que $[SIT]$ est nécessairement réalisable, il est trivial de vérifier qu'il existe un flot sur G_{SIT} saturant l'ensemble des arcs incidents extérieurement à α, c'est-à-dire un flot de valeur $\sum_{j \in J} \sum_{k \in K} \bar{x}^*_{jk}$; ce n'est en effet qu'à

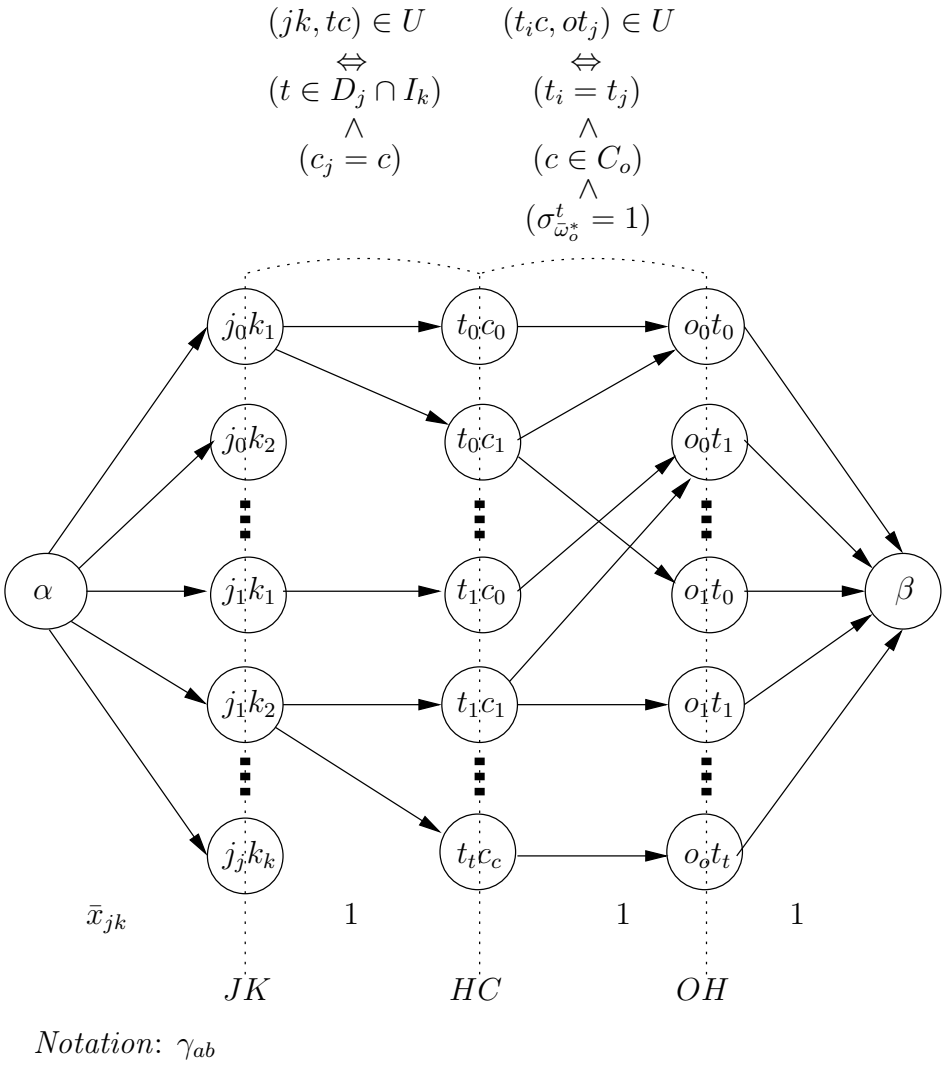

FIGURE 1.2 – Structure du graphe G_{SIT}

cette condition que l'ensemble des niveaux de production induits par la solution $\bar{x}^*$ de $[P]$ sont tous exactement respectés.

Un tel flot peut être obtenu, en un temps polynomial, en utilisant l'algorithme de préflot (*Preflow-Push algorithm*) décrit dans [AMO93].

Une solution complète indexée sur le temps est alors obtenue en utilisant les relations suivantes :

- $\forall j \in J, \forall t \in H$ $x_{jt} = 1$ si la valeur du flot de l'arc (jk, tc_j) (avec $k \in K / t \in I_k$) est égale à 1, 0 sinon.
- $\forall o \in O, \forall c \in C_o, \forall t \in H$ $z_{oct} = 1$ si la valeur du flot de l'arc (tc, ot) est égale à 1, 0 sinon.

Chapitre 2

Bornes inférieures par décomposition lagrangienne

Afin d'obtenir des bornes inférieures à notre problème, nous avons étudié deux décompositions lagrangiennes s'appuyant toutes deux sur la formalisation $[P]$ proposée en 1.2.4. L'idée sous-jacente de ces relaxations est de découpler les deux niveaux de décision de notre problématique, à savoir la planification des emplois du temps et l'ordonnancement des jobs. Nous présentons dans un premier temps une décomposition lagrangienne dualisant l'ensemble des contraintes couplantes, i.e. (1.12) et (1.13). Une seconde relaxation lagrangienne dualisant uniquement (1.13) est ensuite étudiée.

2.1 Décomposition lagrangienne *totale*

Dans cette première approche, nous dualisons les deux jeux de contraintes couplantes (1.12) et (1.13) du modèle $[P]$ proposé en 1.2.4. L'objectif de ce procédé est d'obtenir rapidement, du fait de la simplicité a priori du problème relaxé, une borne inférieure au problème global.

En dualisant (1.12) et (1.13), on obtient la fonction de Lagrange suivante :

$$\begin{aligned}\mathcal{L}_{tot}(x,y,z,\tau,\pi) = & \sum_{o\in O}\sum_{\omega\in\Omega_o} \eta^o_\omega \cdot y^o_\omega + \\ & \sum_{k\in K}\sum_{c\in C} \tau_{kc} \cdot \left(\sum_{\substack{j\in J| \\ c_j=c}} x_{jk} - \sum_{\substack{\theta\in\Theta| \\ c\in\theta}} z_{\theta ck} \right) + \\ & \sum_{k\in K}\sum_{\theta\in\Theta} \pi_{k\theta} \cdot \left(\sum_{c\in\theta} z_{\theta ck} - \sum_{\substack{o\in O| \\ \theta_o=\theta}} \sum_{\substack{\omega\in\Omega_o| \\ I_k\subseteq\omega}} l_k \cdot y^o_\omega \right)\end{aligned}$$

avec

- $\forall k \in K$, $\forall c \in C$, $\tau_{kc} \in \mathbb{R}$ le multiplicateur de Lagrange de chaque contrainte (1.12)
- $\forall k \in K$, $\theta \in \Theta$, $\pi_{k\theta} \in \mathbb{R}^+$ le multiplicateur de Lagrange de chaque contrainte (1.13)

La fonction duale associée à toute distribution (τ, π) des multiplicateurs de Lagrange est alors :

$$
\begin{aligned}
L_{tot}(\tau,\pi) = \min \quad & \mathcal{L}_{tot}(x,y,z,\tau,\pi) & & \\
\forall o \in O \quad & \sum_{\omega \in \Omega_o} y^o_\omega & = 1 & \\
\forall j \in J \quad & \sum_{\substack{k \in K| \\ I_k \subseteq D_j}} x_{jk} & = p_j & \\
& \sum_{\theta \in \Theta} \sum_{c \in \theta} \sum_{k \in K} z_{\theta ck} & = \sum_{j \in J} p_j & \quad (2.1) \\
\forall o \in O, \forall \omega \in \Omega_o \quad & y^o_\omega \in \{0,1\} & & \\
\forall j \in J, \forall k \in K \quad & x_{jk} \in [\![0, \min(p_j, l_k)]\!] & & \\
\forall \theta \in \Theta, \forall c \in \theta, \forall k \in K \quad & z_{\theta ck} \in [\![0, l_k \cdot \text{card}\,\{o \in O|_{\theta_o = \theta}\}]\!] & &
\end{aligned}
$$

On note ici l'ajout, par rapport à la formalisation $[P]$, d'un nouveau groupe de contraintes : (2.1). (2.1) implique l'égalité entre le nombre total de compétences utilisées et la somme des durées des jobs. C'est une contrainte *redondante* pour $[P]$; il en effet trivial de constater que respecter (1.12) conduit nécessairement à vérifier (2.1). Néanmoins, du fait de la dualisation de (1.12), plus rien n'oblige ici le respect de cette contrainte non couplante. Nous ajoutons par conséquent l'égalité (2.1) afin de moins dénaturer le problème relaxé étudié.

Ce Programme Linéaire en Nombres Entiers se décompose en trois sous-problèmes indépendants. On a $L_{tot}(\tau,\pi) = L^x_{tot}(\tau) + L^y_{tot}(\pi) + L^z_{tot}(\tau,\pi)$, avec

$$
\begin{aligned}
L^x_{tot}(\tau) = \min \quad & \sum_{k \in K} \sum_{c \in C} \sum_{\substack{j \in J| \\ c_j = c}} x_{jk} \cdot \tau_{kc} & & \\
\forall j \in J \quad & \sum_{\substack{k \in K| \\ I_k \subseteq D_j}} x_{jk} & = p_j & \\
\forall j \in J, \forall k \in K \quad & x_{jk} \in [\![0, \min(p_j, l_k)]\!] & &
\end{aligned}
$$

$$
\begin{aligned}
L^y_{tot}(\pi) = \min \quad & \sum_{o\in O}\sum_{\omega\in\Omega_o} y^o_\omega \cdot \left(\eta^o_\omega + \sum_{\substack{k\in K| \\ I_k\subseteq\omega}} \pi_{k\theta_o} \cdot l_k \right) \\
\forall o \in O \qquad & \sum_{\omega\in\Omega_o} y^o_\omega = 1 \\
\forall o \in O, \forall \omega \in \Omega_o \qquad & y^o_\omega \in \{0,1\}
\end{aligned}
$$

et

$$
\begin{aligned}
L^z_{tot}(\tau,\pi) = \min \quad & \sum_{\theta\in\Theta}\sum_{c\in\theta}\sum_{k\in K} z_{\theta ck} \cdot (\pi_{k\theta} - \tau_{kc}) \\
& \sum_{\theta\in\Theta}\sum_{c\in\theta}\sum_{k\in K} z_{\theta ck} = \sum_{j\in J} p_j \\
\forall \theta \in \Theta, \forall c \in \theta, \forall k \in K \quad & z_{\theta ck} \in [\![0, l_k \cdot \mathrm{card}\,\{o \in O|_{\theta_o=\theta}\}]\!]
\end{aligned}
$$

Pour une distribution de τ et π fixée, la résolution de ces trois sous-problèmes est polynomiale.

Calculer $L^x_{tot}(\tau)$ consiste à ordonnancer l'ensemble des jobs au moindre coût, τ_{kc} étant le coût de réalisation d'une unité de job requérant la compétence c sur l'intervalle I_k. Le calcul de $L^x_{tot}(\tau)$ se fait alors en sélectionnant les couples (période de temps k - compétence c) par ordre croissant de leur multiplicateur τ_{kc}. On ordonnance ensuite, pour chaque job requérant la compétence c, autant d'unités que possible sur l'intervalle de temps I_k. Le calcul s'arrête lorsque tous les jobs ont été entièrement ordonnancés. L'algorithme 1 détaille les étapes de ce calcul.

Le calcul de $L^y_{tot}(\pi)$ est réalisé en sélectionnant le roulement $\omega \in \Omega_o$ de moindre coût régularisé $(\eta^o_\omega + \sum_{k\in K/I_k\subseteq\omega} \pi_{k\theta_o} \cdot l_k)$ pour chaque opérateur $o \in O$. Les grandes lignes du calcul de $L^y_{tot}(\pi)$ sont données par l'algorithme 2.

Le calcul de $L^z_{tot}(\tau,\pi)$ se fait en sélectionnant les variables $z_{\theta ck}$ par ordre croissant de leur coût régularisé $(\pi_{k\theta} - \tau_{kc})$, puis en les saturant jusqu'à ce que leur somme vaille la somme des durées des jobs. L'algorithme 3 résume la démarche du calcul de $L^z_{tot}(\tau,\pi)$ décrite ici.

Algorithme 1 Calcul de $L^x_{tot}(\tau)$

$z \leftarrow 0$; $J^* \leftarrow J$
$\forall j \in J \quad \gamma_j \leftarrow p_j$
$L \leftarrow$ Liste des multiplicateurs τ_{kc} triés par ordre croissant
Répéter
  $\tau^*_{kc} \leftarrow 1^{er}$ élément de L
  Pour tout $j \in J^*$ **Faire**
    Si $(c_j = c) \wedge (D_j \cap I_k \neq \emptyset)$ **alors**
      $x_{jk} \leftarrow \min(\gamma_j, \text{card}\,\{D_j \cap I_k\})$
      $\gamma_j \leftarrow \gamma_j - x_{jk}$
      $z \leftarrow z + \tau^*_{kc} \cdot x_{jk}$
    Fin Si
    Si $\gamma_j = 0$ **alors**
      $J^* \leftarrow J^* \backslash \{j\}$
    Fin Si
  Fin Pour
  $L \leftarrow L \backslash \{\tau^*_{kc}\}$
Jusqu'à $J^* = \emptyset$
Renvoyer z

Algorithme 2 Calcul de $L^y_{tot}(\pi)$

$z \leftarrow 0$
Pour tout $o \in O$ **Faire**
  $\min \leftarrow \infty$
  Pour tout $\omega \in \Omega_o$ **Faire**
    Si $\left(\eta^o_\omega + \sum_{k \in K / I_k \subseteq \omega} \pi_{k\theta_o} \cdot l_k < \min \right)$ **alors**
      $\min \leftarrow \eta^o_\omega + \sum_{k \in K / I_k \subseteq \omega} \pi_{k\theta_o} \cdot l_k$
      $\omega^* \leftarrow \omega$
    Fin Si
  Fin Pour
  $y^o_{\omega^*} \leftarrow 1$
  $z \leftarrow z + \min$
Fin Pour
Renvoyer z

Algorithme 3 Calcul de $L^z_{tot}(\tau, \pi)$

$z \leftarrow 0$; somme $\leftarrow \sum_{j \in J} p_j$
$L \leftarrow$ Liste des coûts régularisés $\alpha_{\theta kc}$ (avec $\alpha_{\theta kc} = \pi_{k\theta} - \tau_{kc}$) triés par ordre croissant
Répéter
  $\alpha^*_{\theta kc} \leftarrow 1^{er}$ élément de L
  $z_{\theta ck} \leftarrow \min(\text{somme}, l_k \cdot \text{card}\{o \in O|_{\theta_o = \theta}\})$
  somme $\leftarrow$ somme $- z_{\theta ck}$
  $z \leftarrow z + z_{\theta ck} \cdot \alpha^*_{\theta kc}$
  $L \leftarrow L \backslash \{\alpha^*_{\theta kc}\}$
Jusqu'à somme $= 0$
Renvoyer z

Le problème dual consiste à résoudre :

$$[D] : L_{tot}(\tau^*, \pi^*) = \max_{\tau, \pi} L_{tot}(\tau, \pi)$$

D'après le théorême de la dualité faible, $[P]$ étant un problème en minimisation, $L_{tot}(\tau^*, \pi^*)$ est une borne inférieure de la valeur optimale de $[P]$.

La maximisation de L_{tot} peut être effectuée grâce à une méthode de sous-gradient classique [HWC74].

Les trois-sous problèmes lagrangiens décrits ici possèdent la propriété d'intégrité ; les méthodes présentées ci-dessus permettent d'ailleurs toutes d'obtenir des solutions optimales entières. La valeur de la borne calculée est donc équivalente à la borne obtenue en résolvant la relaxation continue de $[P]$ [Geo74]. Notons que la décomposition lagrangienne totale définie ici se calcule sans solveur de programmation linéaire ; il est donc théoriquement possible de s'affranchir d'un tel outil pour atteindre une borne de qualité identique à la relaxation continue.

Cette première relaxation lagrangienne a pour intérêt d'avoir trois sous-problèmes simples à résoudre. Cependant, la dualisation des deux jeux de contraintes couplantes (1.12) et (1.13) induit un problème relaxé assez dénaturé. La complexité du problème se trouve alors transférée vers la méthode de sous-gradient. On peut ainsi naturellement présager d'une certaine lenteur de convergence de la technique. Dans cette perspective, nous définissons ci-après une nouvelle décomposition lagrangienne, moins drastique quant à la dénaturation du problème global.

2.2 Décomposition lagrangienne *partielle*

Le calcul de la seconde borne inférieure par décomposition lagrangienne s'appuie là encore sur la formalisation $[P]$ décrite en 1.2.4. La contrainte couplante (1.13) est ici la seule contrainte dualisée. La fonction de Lagrange alors obtenue est :

$$\mathcal{L}_{part}(x,y,z,\pi) = \sum_{o\in O}\sum_{\omega\in\Omega_o} \eta^o_\omega \cdot y^o_\omega + \sum_{k\in K}\sum_{\theta\in\Theta} \pi_{k\theta} \cdot \left(\sum_{c\in\theta} z_{\theta ck} - \sum_{\substack{o\in O|\\ \theta_o=\theta}} \sum_{\substack{\omega\in\Omega_o|\\ I_k\subseteq\omega}} l_k \cdot y^o_\omega \right)$$

avec $\forall k \in K, \forall \theta \in \Theta \quad \pi_{k\theta} \in \mathbb{R}^+$ le multiplicateur de Lagrange de chaque contrainte (1.13).

La fonction duale associée à toute distribution (π) des multiplicateurs de Lagrange est alors :

$$\begin{array}{rll}
L_{part}(\pi) = \min & \mathcal{L}_{part}(x,y,z,\pi) & \\
\forall o \in O & \sum_{\omega\in\Omega_o} y^o_\omega & = 1 \\
\forall j \in J & \sum_{\substack{k\in K|\\ I_k\subseteq D_j}} x_{jk} & = p_j \\
\forall k \in K, \forall c \in C & \sum_{\substack{j\in J|\\ c_j=c}} x_{jk} - \sum_{\substack{\theta\in\Theta|\\ c\in\theta}} z_{\theta ck} & = 0 \\
\forall k \in K, \forall \theta \in \Theta & \sum_{c\in\theta} z_{\theta ck} & \leq l_k \cdot \mathrm{card}\left\{o \in O|_{\theta_o=\theta}\right\}
\end{array} \tag{2.2}$$

$$\begin{array}{rl}
\forall o \in O, \forall \omega \in \Omega_o & y^o_\omega \in \{0,1\} \\
\forall j \in J, \forall k \in K & x_{jk} \in [\![0, \min(p_j, l_k)]\!] \\
\forall \theta \in \Theta, \forall c \in \theta, \forall k \in K & z_{\theta ck} \in [\![0, l_k \cdot \mathrm{card}\left\{o \in O|_{\theta_o=\theta}\right\}]\!]
\end{array}$$

À l'instar de la première décomposition lagrangienne, on introduit ici aussi un groupe de contraintes redondantes pour $[P]$: (2.2). La contrainte non couplante (2.2) limite le nombre d'employés pouvant travailler sur une machine durant un intervalle de temps donné. Dans le modèle complet $[P]$ présenté en 1.2.4, il est trivial de vérifier que le respect de (1.13) implique le respect de (2.2). Dans la mesure où (1.13) est ici relaxée, nous ajoutons (2.2) afin, là encore, de moins dénaturer le problème relaxé étudié.

$\mathcal{L}_{part}(x, y, z, \pi)$ se décompose en deux sous-problèmes indépendants. On a $L_{part}(\pi) = L^y_{part}(\pi) + L^{xz}_{part}(\pi)$, avec

$$
\begin{aligned}
L^y_{part}(\pi) = \min \quad & \sum_{o \in O} \sum_{\omega \in \Omega_o} y^o_\omega \cdot \left(\eta^o_\omega - \sum_{\substack{k \in K | \\ I_k \subseteq \omega}} \pi_{k\theta_o} \cdot l_k \right) & \\
\forall o \in O \quad & \sum_{\omega \in \Omega_o} y^o_\omega & = 1 \\
\forall o \in O, \forall \omega \in \Omega_o \quad & y^o_\omega \in \{0, 1\} &
\end{aligned}
$$

et

$$
\begin{aligned}
L^{xz}_{part}(\pi) = \min \quad & \sum_{k \in K} \sum_{\theta \in \Theta} \sum_{c \in \theta} \pi_{k\theta} \cdot z_{\theta ck} & \\
\forall j \in J \quad & \sum_{\substack{k \in K | \\ I_k \subseteq D_j}} x_{jk} & = p_j \\
\forall k \in K, \forall c \in C \quad & \sum_{\substack{j \in J | \\ c_j = c}} x_{jk} - \sum_{\substack{\theta \in \Theta | \\ c \in \theta}} z_{\theta ck} & = 0 \\
\forall k \in K, \forall \theta \in \Theta \quad & \sum_{c \in \theta} z_{\theta ck} & \leq l_k \cdot \operatorname{card} \{o \in O|_{\theta_o = \theta}\} \\
\forall j \in J, \forall k \in K \quad & x_{jk} \in [\![0, \min(p_j, l_k)]\!] & \\
\forall \theta \in \Theta, \forall c \in \theta, \forall k \in K \quad & z_{\theta ck} \in [\![0, l_k \cdot \operatorname{card} \{o \in O|_{\theta_o = \theta}\}]\!] &
\end{aligned}
$$

Pour une distribution de π fixée, la résolution de ces deux sous-problèmes est polynomiale.

$L^y_{part}(\pi)$ est équivalent à $L^y_{tot}(\pi)$. $L^y_{part}(\pi)$ se calcule alors en appliquant l'algorithme 2, c'est-à-dire en sélectionnant le roulement de moindre coût régularisé $(\eta^o_\omega + \sum_{k \in K / I_k \subseteq \omega} \pi_{k\theta_o} \cdot l_k)$ pour chaque opérateur $o \in O$.

$L^{xz}_{part}(\pi)$ est l'expression d'un problème de *flot maximum à coût minimum* sur un graphe orienté décomposable en niveaux ; notons le $G_{lag} = (X, U)$, avec

- Ensemble des noeuds : $X = \{s\} \cup J \cup KC \cup \Theta K \cup \{t\}$
  - s : source
  - J : Ensemble des jobs J
  - KC : Produit cartésien de K et C
  - ΘK : Produit cartésien de Θ et K
  - t : puits

- Ensemble des arcs : $\{\alpha, \beta\} \in U$ - (capacité $\gamma_{\alpha\beta}$, coût $\mu_{\alpha\beta}$) -
  - $\forall j \in J : (s, j) \in U$, avec $\gamma_{sj} = p_j$ et $\mu_{sj} = 0$
  - $\forall j \in J, \forall a \in KC : (j, a) \in U \Leftrightarrow (c_j = c_a) \wedge (I_{k_a} \subseteq D_j)$, avec $\gamma_{ja} = \min(p_j, l_{k_a})$ et $\mu_{ja} = 0$
  - $\forall a \in KC, \forall b \in \Theta K : (a, b) \in U \Leftrightarrow (I_{k_a} = I_{k_b}) \wedge (c_a \in \theta_b)$, avec $\gamma_{ab} = l_{k_a} \cdot \text{card}\{o \in O|_{\theta_o = \theta_b}\}$ et $\mu_{ab} = \pi_{k\theta_b}$
  - $\forall b \in \Theta K : (b, t) \in U$, avec $\gamma_{bt} = l_{k_b} \cdot \text{card}\{o \in O|_{\theta_o = \theta_b}\}$ et $\mu_{bt} = 0$

La figure 2.1 décrit la structure du graphe G_{lag}.

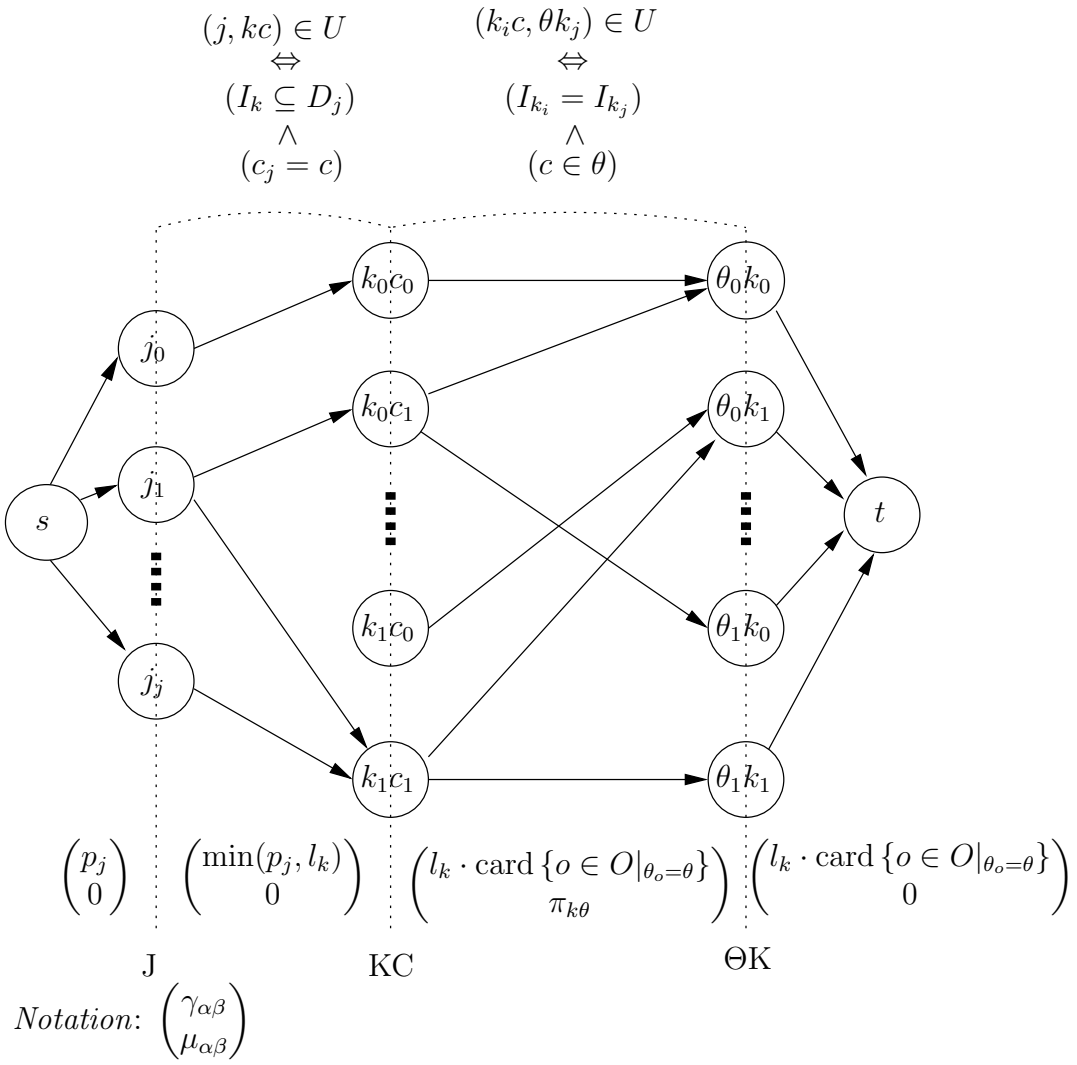

FIGURE 2.1 – Structure de G_{lag}

Le calcul de $L^{xz}_{part}(\pi)$ consiste à déterminer un flot maximum à coût minimum sur G_{lag}.

Proposition 1 (Valeur du flot maximal sur G_{lag})
La valeur du flot maximal sur G_{lag} est égale à la somme des durées des jobs de J, soit $\sum_{j\in J} p_j$.

Preuve.
Supposont un flot F_{lag} saturant l'ensemble des arcs incidents extérieurement à s.

Le débit de F_{lag} est égal à la somme des capacités des arcs (s, j) (avec $j \in J$), soit $\sum_{j\in J} p_j$. $\sum_{j\in J} p_j$ est donc une borne supérieure au flot maximal sur G_{lag}.

Vérifions désormais que F_{lag} est compatible à G_{lag}, c'est-à-dire vérifions que la loi de conservation aux noeuds est bien respectée en tout sommet de G_{lag}.

Il est trivial de vérifier que, par construction du graphe G_{lag}, à chaque sommet $j \in J$ correspond suffisamment de sommets $kc \in KC$ incidents pour que le flot entrant en j puisse être transféré vers ses successeurs.

La réalisabilité de F_{lag} est alors conditionnée par le respect de la loi de conservation aux noeuds des sommets $kc \in KC$ et $\theta k \in \Theta K$.

La capacité des arcs incidents extérieurement à chaque sommet $kc \in KC$ et $\theta k \in \Theta K$ est égale à $l_k \cdot \text{card}\{o \in O|_{\theta_o=\theta}\}$. Cela revient ainsi à considérer, concrètement, que chaque opérateur $o \in O$ puisse travailler à tout instant durant chaque intervalle de temps I_k (ce qui correspond bien à la dualisation en cours ici des contraintes de ressources (1.13)). Les ressources en compétences des opérateurs sont alors supposées ici limitées uniquement par le nombre d'opérateurs impliqués dans le problème.

Dès lors, on voit clairement que la loi de conservation aux noeuds en chaque sommet $kc \in KC$ et $\theta k \in \Theta K$ est respectée si le nombre m d'opérateurs est suffisamment élevé ; une condition suffisante pour cela est que $[P]$ soit réalisable.

Par conséquent, si on suppose qu'il existe au moins une solution réalisable pour $[P]$, la loi de conservation en chaque noeud de G_{lag} est respectée, et F_{lag} est un flot compatible de valeur $\sum_{j\in J} p_j$. ■

Le problème dual consiste à résoudre :

$$[D] : L_{part}(\pi^*) = \max_{\pi} L_{part}(\pi)$$

D'après le théorême de la dualité faible, $L_{part}(\pi^*)$ est une borne inférieure de la valeur optimale de $[P]$.

La maximisation de L_{part} peut être effectuée grâce à une méthode de sous-gradient classique [HWC74].

Tout comme la première relaxation lagrangienne décrite ci-dessus, les deux sous-problèmes de cette seconde décomposition possèdent la propriété d'intégrité ; les méthodes présentées ci-dessus permettent d'ailleurs toutes deux d'obtenir des solutions

optimales entières. La valeur de la borne calculée est donc équivalente à la borne obtenue en résolvant la relaxation continue de $[P]$ [Geo74]. Là encore, la décomposition lagrangienne proposée ici se calcule sans solveur de programmation linéaire ; il est donc théoriquement possible de s'affranchir d'un tel outil pour atteindre une borne de qualité identique à la relaxation continue.

Chapitre 3

Décomposition de Benders

Dans ce chapitre, nous nous intéressons à une première méthode exacte pour résoudre notre problème couplant planification d'agents et ordonnancement de production. La technique décrite ici exploite la structure en deux niveaux de décision du modèle $[P]$ décrit en 1.2.4. Il semble en effet assez naturel, devant une telle formalisation en blocs, d'expérimenter une décomposition de Benders [Ben62].

Nous débutons ce chapitre en proposant une nouvelle formalisation pour la problématique générale. Les deux sous-problèmes impliqués dans notre décomposition de Benders sont ensuite détaillés à tour de rôle. Le schéma de résolution général ainsi que l'algorithme de la méthode sont enfin présentés.

3.1 Nouvelle formalisation

Afin d'assurer l'existence d'une solution réalisable bornée à chaque itération de la méthode, nous proposons une nouvelle formalisation $[P']$ pour notre problème couplant planification d'agents et ordonnancement de production. Ce nouveau modèle $[P']$ s'appuie sur la formalisation $[P]$ décrite en 1.2.4. Seul un jeu de variables d'écart $s_{k\theta}$ positives est associé aux contraintes (1.13). $[P']$ s'écrit alors :

$$[P'] : \min \quad \sum_{o\in O}\sum_{\omega\in\Omega_o} \eta_\omega^o \cdot y_\omega^o + M \cdot \left(\sum_{k\in K}\sum_{\theta\in\Theta} s_{k\theta}\right)$$

$$\begin{aligned}
\forall o \in O \quad & \sum_{\omega\in\Omega_o} y_\omega^o && = 1 \qquad (3.1)\\
\forall j \in J \quad & \sum_{\substack{k\in K|\\ I_k\subseteq D_j}} x_{jk} && = p_j\\
\forall k \in K, \forall c \in C \quad & \sum_{\substack{j\in J|\\ c_j=c}} x_{jk} - \sum_{\substack{\theta\in\Theta|\\ c\in\theta}} z_{\theta ck} && = 0\\
\forall k \in K, \forall \theta \in \Theta \quad & \sum_{c\in\theta} z_{\theta ck} - \sum_{\substack{o\in O|\\ \theta_o=\theta}} \sum_{\substack{\omega\in\Omega_o|\\ I_k\subseteq\omega}} l_k \cdot y_\omega^o - s_{k\theta} && \leq 0\\
\forall o \in O, \forall \omega \in \Omega_o \quad & y_\omega^o \in \{0,1\} && \qquad (3.2)\\
\forall j \in J, \forall k \in K \quad & x_{jk} \in [\![0, \min(p_j, l_k)]\!]\\
\forall \theta \in \Theta, \forall c \in \theta, \forall k \in K \quad & z_{\theta ck} \in [\![0, l_k \cdot \mathrm{card}\,\{o \in O|_{\theta_o=\theta}\}]\!]\\
\forall k \in K, \forall \theta \in \Theta \quad & s_{k\theta} \in \mathbb{N}
\end{aligned}$$

Proposition 2 (Équivalence de solutions optimales entre (P) et (P'))
$[P]$ et $[P']$ fournissent la même solution optimale pour tout $M > \sum_{o\in O}\left(\max_{\omega\in\Omega_o}\eta_\omega^o - \min_{\omega\in\Omega_o}\eta_\omega^o\right)$.

Preuve.
$\sum_{o\in O}\max_{\omega\in\Omega_o}\eta_\omega^o$ (solution consistant à sélectionner le roulement admissible de coût le plus élevé pour chaque opérateur) est une borne supérieure triviale de $[P]$.

$\sum_{o\in O}\min_{\omega\in\Omega_o}\eta_\omega^o + M$ doit être une borne inférieure de $[P']$ pour toute solution irréalisable de $[P]$, c'est-à-dire pour toute solution $\{\bar{x}, \bar{y}, \bar{z}, \bar{s}\}$ telle qu'il existe au moins une variable d'écart positive $\bar{s}_{k\theta} \geq 1$. Par conséquent, pour s'assurer la même solution optimale pour $[P]$ et $[P']$, on doit choisir un nombre M tel que $\sum_{o\in O}\max_{\omega\in\Omega_o}\eta_\omega^o < \sum_{o\in O}\min_{\omega\in\Omega_o}\eta_\omega^o + M$. ■

3.2 Définition du sous-problème

Supposons une distribution $\bar{y}$ des roulements aux opérateurs fixée. Le sous-problème résultant de $[P']$ est alors $[SP'(\bar{y})]$, avec :

$$[SP'(\bar{y})] : \min \quad \sum_{k\in K}\sum_{\theta\in\Theta} s_{k\theta}$$

$$\forall j \in J \quad \sum_{\substack{k\in K| \\ I_k\subseteq D_j}} x_{jk} = p_j \tag{3.3}$$

$$\forall k \in K, \forall c \in C \quad \sum_{\substack{j\in J| \\ c_j=c}} x_{jk} - \sum_{\substack{\theta\in\Theta| \\ c\in\theta}} z_{\theta ck} = 0 \tag{3.4}$$

$$\forall k \in K, \forall \theta \in \Theta \quad \sum_{c\in\theta} z_{\theta ck} - s_{k\theta} \leq \sum_{\substack{o\in O| \\ \theta_o=\theta}} \sum_{\substack{\omega\in\Omega_o| \\ I_k\subseteq\omega}} l_k \cdot \bar{y}^o_\omega \tag{3.5}$$

$$\forall j \in J, \forall k \in K \quad x_{jk} \in [\![0, \min(p_j, l_k)]\!]$$

$$\forall \theta \in \Theta, \forall c \in \theta, \forall k \in K \quad z_{\theta ck} \in [\![0, l_k \cdot \text{card}\,\{o \in O|_{\theta_o=\theta}\}]\!]$$

$$\forall k \in K, \forall \theta \in \Theta \quad s_{k\theta}\mathbb{N}$$

Il est facile de vérifier que, pour toute distribution $\bar{y}$, $[SP'(\bar{y})]$ admet nécessairement une solution réalisable optimale bornée (grâce aux variables $s_{k\theta}$).

En particulier, lorsque $[SP'(\bar{y})]$ admet pour optimum une solution de coût nul, chaque variable d'écart $s_{k\theta}$ est nulle. Le second membre de la fonction-objectif de $[P']$ est alors nul et $\bar{y}$ est une solution réalisable du problème original $[P]$.

Proposition 3 (Totale unimodularité de la matrice de $(SP'(\bar{y}))$)
La matrice de contraintes (notons la $A_{\bar{y}}$) de $[SP'(\bar{y})]$ est totalement unimodulaire.

Preuve.
Les conditions suivantes sont toutes respectées :

1. Chaque terme de $A_{\bar{y}}$ vaut -1, 0 ou +1
2. Chaque colonne de $A_{\bar{y}}$ contient au plus deux termes non nuls (trivial pour les variables $z_{\theta ck}$ et $s_{k\theta}$ et vrai pour les variables x_{jk} car chaque job ne requiert qu'une unique compétence)
3. L'ensemble I des lignes de $A_{\bar{y}}$ peut être partitionné en 2 ensembles $I_1 = \{(3.3)\}$ et $I_2 = \{(3.4),(3.5)\}$ tels que
   - deux termes non nuls de même signe d'une colonne de $A_{\bar{y}}$ sont l'un dans une ligne de I_1, l'autre dans une ligne de I_2
   - deux termes non nuls de signe opposé d'une colonne de $A_{\bar{y}}$ sont tous deux dans I_1, ou tous deux dans I_2

■

Proposition 4 (Propriété d'intégrité) *La solution optimale de la relaxation continue de $[SP'(\bar{y})]$ est entière, et correspond alors à la solution optimale de $[SP'(\bar{y})]$.*

Preuve.
D'après la proposition 3, la matrice $A_{\bar{y}}$ des contraintes de $[SP'(\bar{y})]$ est totalement unimodulaire. On sait de plus que les seconds membres de $[SP'(\bar{y})]$ sont tous entiers. L'optimum de la relaxation continue de $[SP'(\bar{y})]$ est donc entier, et coïncide avec l'optimum de $[SP'(\bar{y})]$. ■

On peut donc relaxer les contraintes d'intégrité des variables de $[SP'(\bar{y})]$ et se ramener à l'étude de sa relaxation continue (programme linéaire en variables continues), dont le dual $[DSP'(\bar{y})]$ s'écrit :

$$\begin{aligned}
[DSP'(\bar{y})] : \max \quad & g_{\bar{y}}(a,b,c,d,e) = \sum_{j\in J} a_j \cdot p_j + \\
& \sum_{k\in K}\sum_{\theta\in\Theta} c_{k\theta} \cdot \sum_{\substack{o\in O| \\ \theta_o=\theta}} \sum_{\substack{\omega\in\Omega_o| \\ I_k\subseteq\omega}} l_k \cdot \bar{y}^o_\omega + \\
& \sum_{j\in J}\sum_{k\in K} \min(p_j, l_k)\cdot d_{jk} + \\
& \sum_{\theta\in\Theta}\sum_{c\in\theta}\sum_{k\in K} l_k \cdot \mathrm{card}\left\{o\in O|_{\theta_o=\theta}\right\}\cdot e_{\theta ck}
\end{aligned}$$

$$\begin{aligned}
\forall j\in J, \forall k\in K \quad & \alpha^k_j \cdot a_j + b_{kc_j} + d_j \leq 0 \quad (3.6)\\
\forall\theta\in\Theta, \forall c\in\theta, \forall k\in K \quad & -b_{kc} + c_{k\theta} + e_{\theta ck} \leq 0 \quad (3.7)\\
\forall k\in K, \forall\theta\in\Theta \quad & -c_{k\theta} \leq 1 \quad (3.8)\\
\forall j\in J \quad & a_j \leqslant 0 \quad (3.9)\\
\forall k\in K, \forall c\in C \quad & b_{kc} \leqslant 0 \quad (3.10)\\
\forall k\in K, \forall\theta\in\Theta \quad & c_{k\theta} \leq 0 \quad (3.11)\\
\forall j\in J, \forall k\in K \quad & d_{jk} \leq 0 \quad (3.12)\\
\forall\theta\in\Theta, \forall c\in\theta, \forall k\in K \quad & e_{\theta ck} \leq 0 \quad (3.13)
\end{aligned}$$

où $\alpha^k_j = 1$ si $I_k \subseteq D_j$, $\alpha^k_j = 0$ sinon.

3.3 Expression de (P') en fonction des points extrêmes de $(DSP'(y))$

On peut alors réécrire $[P']$ sous la forme :

$$[P'] : \min_{y/(3.1)\wedge(3.2)} \left\{ \sum_{o\in O}\sum_{\omega\in\Omega_o} \eta^o_\omega \cdot y^o_\omega + M \cdot \min_y SP'(y) \right\}$$

ou encore, en utilisant le théorême de la dualité forte, et en notant D le polyèdre défini par $D = \{(a, b, c, d, e)/(3.6) - (3.13)\}$:

$$[P'] : \min_{y/(3.1)\wedge(3.2)} \left\{ \sum_{o\in O} \sum_{\omega\in\Omega_o} \eta_\omega^o \cdot y_\omega^o + M \cdot \max_{(a,b,c,d,e)\in D} g_y(a, b, c, d, e) \right\}$$

Comme $[SP'(y)]$ admet toujours une solution réalisable bornée, $[DSP'(y)]$ aussi. Ce dernier admet donc au moins un point extrême en lequel son optimum est atteint. Par ailleurs, D ne dépend pas de y. Par conséquent, en considérant l'ensemble Q de ses q points extrêmes $\left(Q = \left\{(\tilde{a}_1, \tilde{b}_1, \tilde{c}_1, \tilde{d}_1, \tilde{e}_1), \ldots, (\tilde{a}_q, \tilde{b}_q, \tilde{c}_q, \tilde{d}_q, \tilde{e}_q)\right\}\right)$, on peut réécrire $[P']$ sous la forme :

$$[P'] : \min_{y/(3.1)\wedge(3.2)} \left\{ \sum_{o\in O} \sum_{\omega\in\Omega_o} \eta_\omega^o \cdot y_\omega^o + M \cdot \max_{i\in[\![1,q]\!]} g_y(\tilde{a}_i, \tilde{b}_i, \tilde{c}_i, \tilde{d}_i, \tilde{e}_i) \right\}$$

3.4 Définition du programme maître

Le problème précédent est équivalent au problème maître $[MP']$:

$$\begin{array}{rll}
[MP'] : \min & \psi & \\
\forall i \in [\![1, q]\!] & \displaystyle\sum_{o\in O} \sum_{\omega\in\Omega_o} \eta_\omega^o \cdot y_\omega^o + M \cdot g_y(\tilde{a}_i, \tilde{b}_i, \tilde{c}_i, \tilde{d}_i, \tilde{e}_i) - \psi & \leq 0 \\
\forall o \in O & \displaystyle\sum_{\omega\in\Omega_o} y_\omega^o & = 1 \\
\forall o \in O, \forall \omega \in \Omega_o & y_\omega^o \in \{0, 1\} & \\
& \psi \in \mathbb{R} &
\end{array}$$

ou encore, sous l'hypothèse que le problème global $[P]$ admette au moins une distribution réalisable y_f de y (auquel cas l'optimum de $[DSP'(y_f)]$ est nul, signifiant alors que $\psi_f = \displaystyle\sum_{o\in O} \sum_{\omega\in\Omega_o} \eta_\omega^o \cdot (y_f)_\omega^o$) :

$$\begin{array}{rll}
[MP'] : \min & \displaystyle\sum_{o\in O} \sum_{\omega\in\Omega_o} \eta_\omega^o \cdot y_\omega^o & \\
\forall i \in [\![1, q]\!] & g_y(\tilde{a}_i, \tilde{b}_i, \tilde{c}_i, \tilde{d}_i, \tilde{e}_i) & \leq 0 \\
\forall o \in O & \displaystyle\sum_{\omega\in\Omega_o} y_\omega^o & = 1 \\
\forall o \in O, \forall \omega \in \Omega_o & y_\omega^o \in \{0, 1\} &
\end{array}$$

3.5 Schéma de résolution

On peut alors résoudre le problème maître $[MP']$ par génération de contraintes (coupes de Benders [Ben62]) : on considère $[RMP'^r]$ le problème déduit de $[MP']$ en n'intégrant qu'un sous-ensemble Q_r de r contraintes :

$$\begin{aligned}
[RMP'^r] : \min \quad & c_{\bar{y}_r} = \sum_{o \in O} \sum_{\omega \in \Omega_o} \eta_\omega^o \cdot y_\omega^o \\
\forall i \in [\![1, r]\!] \quad & g_y(\tilde{a}_i, \tilde{b}_i, \tilde{c}_i, \tilde{d}_i, \tilde{e}_i) \quad \leq 0 \\
\forall o \in O \quad & \sum_{\omega \in \Omega_o} y_\omega^o \quad = 1 \\
\forall o \in O, \forall \omega \in \Omega_o \quad & y_\omega^o \in \{0, 1\}
\end{aligned}$$

Chacun de ces problèmes est résolu avec un solveur de PLNE. Soit $\bar{y}_r$ une solution optimale de $[RMP'^r]$. Pour vérifier que $\bar{y}_r$ est une solution réalisable (et alors optimale) de $[MP']$, il suffit de résoudre le problème $[DSP'(\bar{y}_r)]$ avec un solveur de PLNE. On obtient alors une solution $(\bar{a}_r, \bar{b}_r, \bar{c}_r, \bar{d}_r, \bar{e}_r)$ de coût $\bar{\nu}_r$.

Si $\bar{\nu}_r \leq 0$, alors évidemment

$$\forall i \in [\![1, q]\!] \quad g_{\bar{y}_r}(\tilde{a}_i, \tilde{b}_i, \tilde{c}_i, \tilde{d}_i, \tilde{e}_i) \leq 0$$

et $\bar{y}_r$ est une solution réalisable et optimale de $[MP']$ et donc, par équivalences successives, de $[P']$ et de $[P]$.

Si $\bar{\nu}_r > 0$, alors la contrainte de $[MP']$ liée au point extrême $(\bar{a}_r, \bar{b}_r, \bar{c}_r, \bar{d}_r, \bar{e}_r)$ de D est violée par la solution $\bar{y}_r$. On ajoute donc la contrainte

$$g_y(\bar{a}_r, \bar{b}_r, \bar{c}_r, \bar{d}_r, \bar{e}_r) \leq 0$$

à $[RMP'^r]$ (évidemment pour la première fois puisque que $\bar{y}_r$ est une solution réalisable de $[RMP'^r]$) et on résout le nouveau problème.

Le nombre de points extrêmes du polyèdre D étant fini, le processus converge en un nombre fini d'étapes.

Notons ici que le temps de résolution de chaque problème $[RMP'^r]$ peut être optimisé en récupérant les informations de calcul des étapes précédentes, c'est-à-dire : en appliquant l'algorithme dual du simplexe avec pour base initiale, la base optimale de $[RMP'^{r-1}]$.

3.6 Algorithme

L'algorithme 4 détaille les grandes lignes de la méthode exacte basée sur une décomposition de Benders présentée ci-dessus.

Algorithme 4 Algorithme de la méthode exacte basée sur une décomposition de Benders

$LB \leftarrow 0$, $r \leftarrow 0$
Répéter
  $\bar{y}_r \leftarrow$ solution optimale de $[RMP'^r]$, de coût : $c_{\bar{y}_r}$
  $LB \leftarrow c_{\bar{y}_r}$
  $(\bar{a}_r, \bar{b}_r, \bar{c}_r, \bar{d}_r, \bar{e}_r) \leftarrow$ solution optimale de $[DSP'(\bar{y}_r)]$, de coût : $\bar{\nu}_r$
  Si $\bar{\nu}_r > 0$ **alors**
    ajouter la contrainte $g_y(\bar{a}_r, \bar{b}_r, \bar{c}_r, \bar{d}_r, \bar{e}_r) \leq 0$ à $[RMP'^r]$
  Fin Si
  $r \leftarrow r + 1$
Jusqu'à $(\bar{\nu}_r \leq 0)$
Renvoyer LB

Chapitre 4

Décomposition et génération de coupes

Nous présentons dans ce chapitre une seconde approche exacte de résolution par génération de coupes. À l'instar de la décomposition de Benders présentée au chapitre 3, elle exploite la décomposition naturelle en deux niveaux de décision du problème couplant planification d'agents et ordonnancement de production auquel nous nous intéressons dans cette première partie de la thèse.

Cette procédure de coupes est basée sur la formalisation $[P]$ proposée en 1.2.4. Elle peut être déclinée en deux variantes. Nous débutons ce chapitre en présentant les parties communes aux deux versions, à savoir : leur mécanisme général, les sous-problèmes induits par la décomposition de la problématique générale et le schéma de génération de coupes utilisé. Nous détaillons ensuite les spécificités et algorithme de chacune des deux versions.

4.1 Mécanisme général

Tout comme la décomposition de Benders décrite au chapitre 3, cette méthode exacte exploite la décomposition de $[P]$ (cf. 1.2.4) en deux sous-problèmes. Un problème maître $[MP]$ définit tout d'abord une distribution de roulements aux opérateurs (notons-la $\bar{y}$). Utilisant ces informations comme données, un sous-problème $[SP(\bar{y})]$ vérifie ensuite la réalisabilité, aussi bien au niveau de l'exécution complète des jobs que de la disponibilité suffisante de compétence à chaque instant, de $\bar{y}$. Si $[SP(\bar{y})]$ ne possède aucune solution réalisable, une coupe invalidant $\bar{y}$ est ajoutée à $[MP]$. Le processus itère alors jusqu'à ce que l'affectation de moindre coût conduisant à une solution réalisable dans $[SP(\bar{y})]$ ait été trouvée. Si $[P]$ est irréalisable, le processus itère jusqu'à ce que $[MP]$ n'ait plus de solution réalisable.

4.1.1 Définition du problème maître (MP)

Le problème maître $[MP]$ a pour but d'affecter au moindre coût un roulement à chaque opérateur, les contraintes liant les profils de compétences aux opérateurs

ainsi que celles reliant les jobs aux opérateurs étant relaxées. Une formalisation de $[MP]$ en termes de Programme Linéaire en Nombres Entiers, basée sur le vecteur de variables y et sur les contraintes afférant à ces variables dans la formalisation $[P]$, est par conséquent :

$$\begin{aligned}
[MP] : \min c_y = & \sum_{o \in O} \sum_{\omega \in \Omega_o} \eta_\omega^o \cdot y_\omega^o \\
\forall o \in O \qquad & \sum_{\omega \in \Omega_o} y_\omega^o \qquad = 1 \\
& \textit{Coupes} \\
\forall o \in O, \forall \omega \in \Omega_o \qquad & y_\omega^o \in \{0,1\}
\end{aligned}$$

Coupes désignant l'ensemble des coupes de réalisabilité ajoutées itérativement au modèle. Ces coupes permettent d'invalider des solutions de $[MP]$ ne conduisant pas à une solution réalisable au vu de l'ensemble des contraintes.

4.1.2 Définition du sous-problème $(SP(\bar{y}))$

Considérons une affectation $\bar{y}$ de roulements aux opérateurs comme une solution réalisable de $[MP]$.

Afin de juger de la réalisabilité de $\bar{y}$ pour $[P]$, nous devons vérifier que les contraintes d'exécution complète des jobs et de disponibilité suffisante de compétence à chaque instant peuvent être respectées par $\bar{y}$. Dans cette optique, on introduit le sous-problème $[SP(\bar{y})]$. $[SP(\bar{y})]$ est lui aussi basé sur le modèle $[P]$, il repose sur les contraintes impliquant les variables x et z de ce modèle. Le but de $[SP(\bar{y})]$ est de fournir l'ordonnancement, compatible aux ressources humaines fixées par $\bar{y}$, qui maximise le nombre d'unités de jobs exéctuées. Une modélisation en termes de PLNE de $[SP(\bar{y})]$ est alors :

$$\begin{aligned}
[SP(\bar{y})] : \max \qquad & f_{\bar{y}} = \sum_{j \in J} \sum_{\substack{k \in K| \\ I_k \subseteq D_j}} x_{jk} \\
\forall j \in J \qquad & \sum_{\substack{k \in K| \\ I_k \subseteq D_j}} x_{jk} \qquad \leq p_j \\
\forall k \in K, \forall c \in C \qquad & \sum_{\substack{j \in J| \\ c_j = c}} x_{jk} - \sum_{\substack{\theta \in \Theta| \\ c \in \theta}} z_{\theta ck} \qquad = 0 \\
\forall k \in K, \forall \theta \in \Theta \qquad & \sum_{c \in \theta} z_{\theta ck} \qquad \leq \sum_{\substack{o \in O| \\ \theta_o = \theta}} \sum_{\substack{\omega \in \Omega_o| \\ I_k \subseteq \omega}} l_k \cdot \bar{y}_\omega^o \\
\forall j \in J, \forall k \in K \qquad & x_{jk} \in [\![0, \min(p_j, l_k)]\!] \\
\forall \theta \in \Theta, \forall c \in \theta, \forall k \in K \qquad & z_{\theta ck} \in [\![0, l_k \cdot \operatorname{card}\{o \in O|_{\theta_o = \theta}\}]\!]
\end{aligned}$$

Proposition 5 (Condition nécessaire et suffisante pour la réalisabilité de $\bar{y}$)

$\bar{y}$ est une solution réalisable de $[P]$ si et seulement si la valeur optimale $f^*_{\bar{y}}$ de $[SP(\bar{y})]$ est supérieure ou égale à $\sum_{j\in J} p_j$.

Preuve.
$f^*_{\bar{y}}$ est le nombre maximal d'unités de jobs qu'il est possible d'ordonnancer via $\bar{y}$. Dans la mesure où $[P]$ a pour but d'ordonnancer l'ensemble des jobs entièrement, seule une affectation $\bar{y}$ permettant d'aboutir à une valeur optimale $f^*_{\bar{y}} \geq \sum_{j\in J} p_j$ est réalisable pour $[P]$. Par conséquent, $\bar{y}$ est réalisable si et seulement si $f^*_{\bar{y}} \geq \sum_{j\in J} p_j$. ■

$[SP(\bar{y})]$ est un problème de *flot maximum* sur un graphe orienté décomposable en niveaux ; notons le $G_{\bar{y}} = (X, U)$, avec

- Ensemble des sommets : $X = \{s\} \cup J \cup KC \cup \Theta K \cup \{t\}$
  - s : source
  - J : ensemble des jobs J
  - KC : produit cartésien de K et C
  - ΘK : produit cartésien de Θ et K
  - t : puits
- Ensemble des arcs : $\{\alpha, \beta\} \in U$ - (capacité $\gamma_{\alpha\beta}$) -
  - $\forall j \in J : (s, j) \in U$, avec $\gamma_{sj} = p_j$
  - $\forall j \in J, \forall a \in KC : (j, a) \in U \Leftrightarrow (c_j = c_a) \wedge (I_{k_a} \subseteq D_j)$, avec $\gamma_{ja} = \min(p_j, l_{k_a})$
  - $\forall a \in KC, \forall b \in \Theta K : (a, b) \in U \Leftrightarrow (I_{k_a} = I_{k_b}) \wedge (c_a \in \theta_b)$, avec $\gamma_{ab} = l_{k_b} \cdot \text{card}\{o \in O|_{\theta_o=\theta_b}\}$
  - $\forall b \in \Theta K : (b, t) \in U$, avec $\gamma_{bt} = \sum_{\substack{o\in O| \\ \theta_o=\theta_b}} \sum_{\substack{\omega\in\Omega_o| \\ I_{k_b}\subseteq\omega}} l_{k_b} \cdot \bar{y}^o_\omega$

Les figures 4.1 et 4.2 décrivent respectivement $G_{\bar{y}}$ dans le cas général et pour Exemple 1.

Dans nos expérimentations, nous utilisons l'algorithme de préflot (*Preflow-Push algorithm*) [AMO93] pour résoudre chaque problème de flot maximal $[SP(\bar{y})]$.

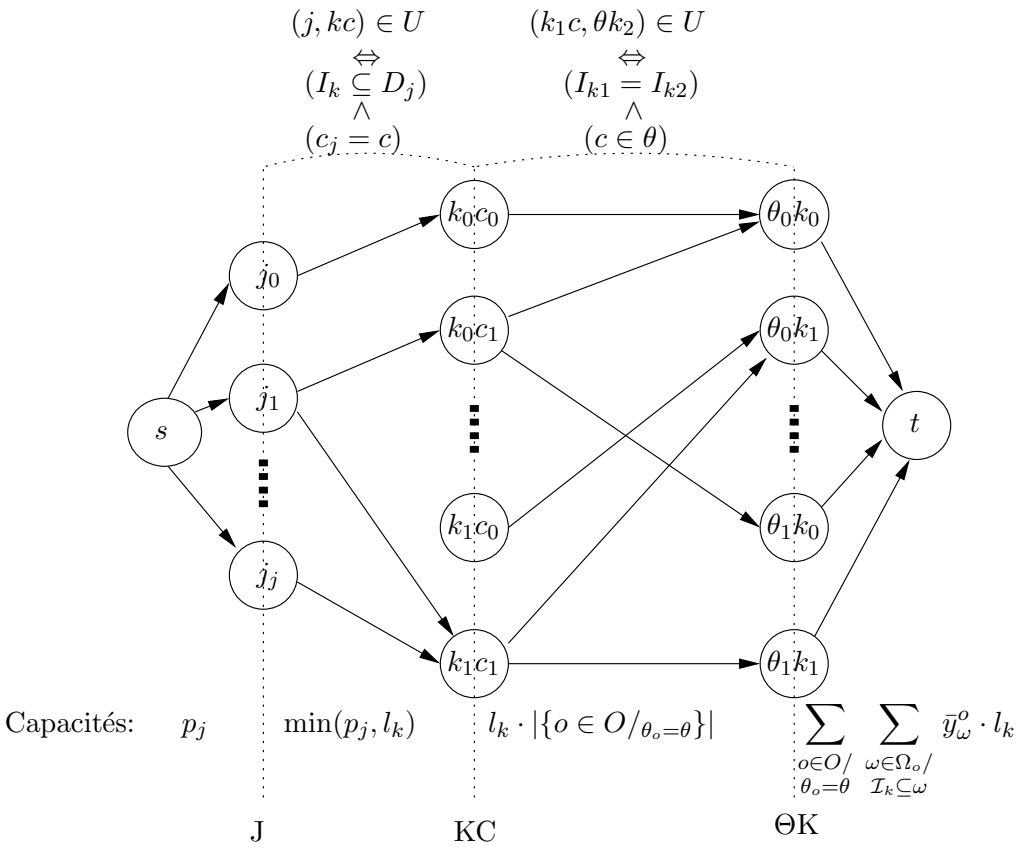

FIGURE 4.1 – Structure du graphe $G_{\bar{y}}$ dans le cas général

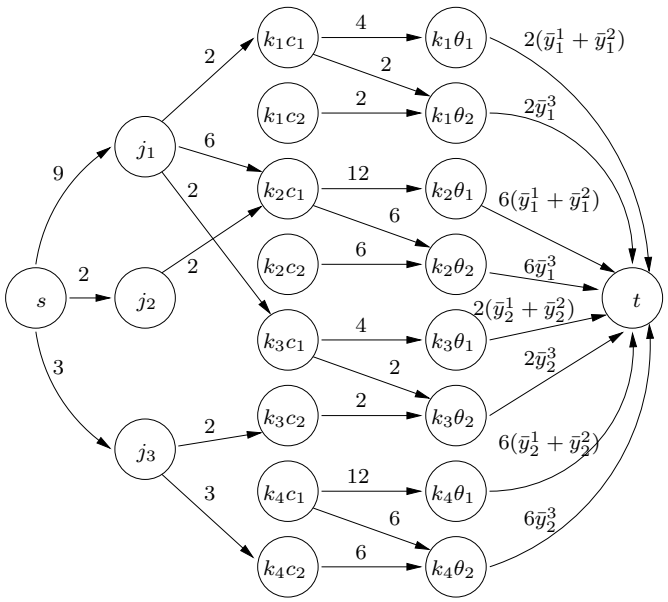

FIGURE 4.2 – Structure du graphe $G_{\bar{y}}$ pour Exemple 1

4.1.3 Schéma de génération de coupes

On a vu, dans la proposition 5, que si la solution optimale de $[SP(\bar{y})]$ a une valeur $f_{\bar{y}}$ inférieure à $\sum_{j\in J} p_j$, il est impossible d'établir un plan d'ordonnancement global avec la distribution $\bar{y}$. $\bar{y}$ doit donc être, le cas échéant, éliminé de l'ensemble des solutions de $[MP]$.

Proposition 6 (Caractérisation d'une solution réalisable pour (P))
Il existe une solution réalisable pour $[P]$ compatible avec $\bar{y}$ si et seulement si :

$$\sum_{b\in\Theta K^+}\sum_{\substack{o\in O|\\ \theta_o=\theta_b}}\sum_{\substack{\omega\in\Omega_o|\\ I_{k_b}\subseteq\omega}} l_{k_b}\cdot\bar{y}^o_\omega \geq \sum_{j\in J^+} p_j - \sum_{j\in J^+}\sum_{a\in KC^-}\min(p_j,l_{k_a}) - \sum_{a\in KC^+}\sum_{b\in\Theta K^-} l_{k_b}\cdot card\{o\in O|_{\theta_o=\theta_b}\}$$

- $\forall u\in U \quad (\phi_u,\gamma_u)$ *: (valeur du flot, capacité) de l'arc* u
- $X = X^+ \cup X^-$ *avec*
  - $X^+ = \{s\} \cup J^+ \cup KC^+ \cup \Theta K^+$
  - $X^- = \overline{X^+} = J^- \cup KC^- \cup \Theta K^- \cup \{t\}$
  - $\forall u=(\alpha,\beta)\in U|_{\{(\alpha\in X^+)\wedge(\beta\in X^-)\}}$ *:* $\phi_u = \gamma_u$
  - $\forall u=(\alpha,\beta)\in U|_{\{(\alpha\in X^-)\wedge(\beta\in X^+)\}}$ *:* $\phi_u = 0$

Preuve.
Considérons un flot $F_{\bar{y}}$ de valeur $f_{\bar{y}}$ sur $G_{\bar{y}}$.

En exploitant la décomposition en niveaux naturelle de $G_{\bar{y}}$ et utilisant les notations et conditions mentionnées ci-dessus, on peut voir que l'ensemble des sommets de $G_{\bar{y}}$ est partitionné en deux sous-ensembles X^+ et X^- contenant respectivement la source s et le puits t. X^+/X^- est alors une coupe (notons-la Ξ) source-puits valide. La figure 4.3 illustre Ξ.

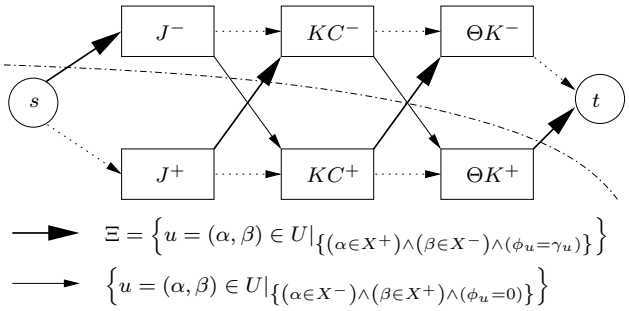

FIGURE 4.3 – Coupe Ξ sur $G_{\bar{y}}$

La capacité de Ξ est, par définition, égale à la somme des capacités des arcs incidents extérieurement à X^+ ($s\in X^+$). La capacité ξ de Ξ vaut ainsi :

$$\xi = \sum_{j\in J^-}\gamma_{sj} + \sum_{j\in J^+}\sum_{a\in KC^-}\gamma_{ja} + \sum_{a\in KC^+}\sum_{b\in\Theta K^-}\gamma_{ab} + \sum_{b\in\Theta K^+}\gamma_{bt}$$

Soit, en remplaçant les capacités par leur valeur :

$$\xi = \sum_{j\in J^-} p_j + \sum_{j\in J^+}\sum_{a\in KC^-} \min(p_j, l_{k_a}) + \sum_{a\in KC^+}\sum_{b\in\Theta K^-} l_{k_b}\cdot\text{card}\,\{o\in O|_{\theta_o=\theta_b}\} + \sum_{b\in\Theta K^+}\sum_{\substack{\omega\in\Omega_o|\\ I_{k_b}\subseteq\omega}} l_{k_b}\cdot\bar{y}^o_\omega$$

Par construction, l'ensemble des arcs incidents extérieurement à X^+ sont saturés. La valeur du flot $f_{\bar{y}}$ est donc égale à la capacité ξ de la coupe Ξ. Par conséquent, par application du théorême de Ford-Fulkerson de *coupe minimale/flot maximal* [FF62], $f_{\bar{y}}$ est la valeur de flot maximal (notons-la $f^*_{\bar{y}}$) sur $G_{\bar{y}}$. On a donc :

$$f^*_{\bar{y}} = \xi$$

On sait d'autre part que $\bar{y}$ est réalisable si et seulement si $f^*_{\bar{y}} \geq \sum_{j\in J} p_j$ (proposition 5). $\bar{y}$ est donc, par équivalence, réalisable si et seulement si :

$$\xi \geq \sum_{j\in J} p_j$$

Soit, en remplaçant ξ par sa valeur :

$$\sum_{b\in\Theta K^+}\sum_{\substack{o\in O|\\ \theta_o=\theta_b}}\sum_{\substack{\omega\in\Omega_o|\\ I_{k_b}\subseteq\omega}} l_{k_b}\cdot\bar{y}^o_\omega \geq \nu$$

$$\text{avec } \nu = \sum_{j\in J^+} p_j - \sum_{j\in J^+}\sum_{a\in KC^-}\min(p_j, l_{k_a}) - \sum_{a\in KC^+}\sum_{b\in\Theta K^-} l_{k_b}\cdot\text{card}\{o\in O|_{\theta_o=\theta_b}\}$$ ■

À chaque solution $\bar{y}$ jugée irréalisable pour $[P]$ par $[SP(\bar{y})]$, on ajoute par conséquent à l'ensemble *Coupes* de $[MP]$ la coupe minimale de flot suivante :

$$\sum_{o\in O}\sum_{\omega\in\Omega_o}\alpha_{o\omega}\cdot y^o_\omega \geq \nu \tag{4.1}$$

où $\forall o\in O$, $\forall\omega\in\Omega_o$, $\alpha_{o\omega} = \sum_{\substack{b\in\Theta K^+|\\ (\theta_b=\theta_p)\wedge(I_{k_b}\subseteq\omega)}} l_{k_b}$

4.1.4 Nature du problème maître

La formalisation en PLNE d'un problème maître restreint $[MP^r]$ avec un sous-ensemble réduit Q_r de r coupes est alors :

$$[MP^r] : \min c_y = \sum_{o\in O}\sum_{\omega\in\Omega_o} \eta^o_\omega \cdot y^o_\omega \tag{4.2}$$

$$\forall o \in O \quad \sum_{\omega\in\Omega_o} y^o_\omega = 1 \tag{4.3}$$

$$\forall i \in [\![1,r]\!] \quad \sum_{o\in O}\sum_{\omega\in\Omega_o} \alpha^i_{o\omega} \cdot y^o_\omega \geq \nu_i \tag{4.4}$$

$$\forall o \in O, \forall \omega \in \Omega_o \quad y^o_\omega \in \{0,1\}$$

Proposition 7 (Identification et complexité de (MP^r))
$[MP^r]$ *est un problème de* sac à dos multi-choix multidimensionnel *(NP-difficile [MT87, MT90]).*

Preuve.
Soit $\bar{\alpha}^i_o = \max_{\omega\in\Omega_o} \alpha^i_{o\omega}$

On a alors (4.4) $\Leftrightarrow \sum_{o\in O} \bar{\alpha}^i_o - \sum_{o\in O}\sum_{\omega\in\Omega_o} \alpha^i_{o\omega} \cdot y^o_\omega \leq \sum_{o\in O} \bar{\alpha}^i_o - \nu_i$, $\forall i \in [\![1,r]\!]$

Or, d'après (4.3) $\sum_{\omega\in\Omega_o} y^o_\omega = 1$, $\forall o \in O$

D'où (4.4) $\Leftrightarrow \sum_{o\in O} \bar{\alpha}^i_o \cdot \sum_{\omega\in\Omega_o} y^o_\omega - \sum_{o\in O}\sum_{\omega\in\Omega_o} \alpha^i_{o\omega} \cdot y^o_\omega \leq \sum_{o\in O} \bar{\alpha}^i_o - \nu_i$, $\forall i \in [\![1,r]\!]$

Soit (4.4) $\Leftrightarrow \sum_{o\in O}\sum_{\omega\in\Omega_o} \left(\bar{\alpha}^i_o - \alpha^i_{o\omega}\right) \cdot y^o_\omega \leq \sum_{o\in O} \bar{\alpha}^i_o - \nu_i$, $\forall i \in [\![1,r]\!]$

avec (par définition de $\bar{\alpha}^i_o$) $\bar{\alpha}^i_o - \alpha^i_{o\omega} \geq 0$, $\forall o \in O$ $\forall \omega \in \Omega_o$ $\forall i \in [\![1,r]\!]$

De la même manière, on prouve que (4.2) est équivalent, à une constante près, à $\max \sum_{o\in O}\sum_{\omega\in\Omega_o} (\bar{\eta}^o - \eta^o_\omega) \cdot y^o_\omega$, avec $\bar{\eta}^o = \max_{\omega\in\Omega_o} \eta^o_\omega$.

$[MP^r]$ est donc, à une constante près, équivalent au problème classique de *Sac à Dos Multi-Choix Multidimensionnel* $[MMKP^r]$ suivant :

$$[MMKP^r] : \max \sum_{o\in O}\sum_{\omega\in\Omega_o} (\bar{\eta}^o - \eta^o_\omega) \cdot y^o_\omega$$

$$\forall o \in O \quad \sum_{\omega\in\Omega_o} y^o_\omega = 1$$

$$\forall i \in [\![1,r]\!] \quad \sum_{o\in O}\sum_{\omega\in\Omega_o} \left(\bar{\alpha}^i_o - \alpha^i_{o\omega}\right) \cdot y^o_\omega \leq \sum_{o\in O} \bar{\alpha}^i_o - \nu_i$$

$$\forall o \in O, \forall \omega \in \Omega_o \quad y^o_\omega \in \{0,1\}$$

■

4.2 Résolution du problème maître

La technique de génération de coupes présentée ci-dessus peut être déclinée en deux variantes ; la différence entre les deux reposant sur la manière de résoudre chaque problème maître $[MP^r]$. Nous détaillons ici les spécificités et justifions l'intérêt de chacune des deux versions.

4.2.1 Résolution exacte

Dans une première version, on se propose de résoudre optimalement chaque problème maître $[MP^r]$. Dès lors, $[MP^r]$ fournissant, à chaque itération, l'affectation réalisable de roulement aux opérateurs de moindre coût, la première affectation $\bar{y}_r$ conduisant à une solution réalisable dans $[SP(\bar{y}_r)]$ est optimale ; le processus est alors arrêté. L'algorithme 5 décrit les grandes étapes de la méthode exacte de génération de coupes lorsque $[MP]$ est résolu optimalement.

Algorithme 5 Variante *exacte* de la méthode de génération de coupes

$LB \leftarrow 0$, $r \leftarrow 0$
Répéter
  $\bar{y}_r \leftarrow$ solution optimale de $[MP^r]$, de coût : $c_{\bar{y}_r}$
  Si $\bar{y}$ est définie **alors**
    $LB \leftarrow c_{\bar{y}_r}$, $r \leftarrow r + 1$
    Si $f_{\bar{y}_r} < \sum_{j \in J} p_j$ **alors**
      ajouter la *coupe minimale de flot* (4.1) sur $G_{\bar{y}_r}$ à $[MP^r]$
    Fin Si
  Fin Si
Jusqu'à $(f_{\bar{y}_r} = \sum_{j \in J} p_j) \vee (\bar{y}_r$ n'est pas définie)
Renvoyer LB

Dans nos expérimentations, $[MP^r]$ est résolu via un solveur de PLNE (Ilog Cplex 9.1).

L'intérêt de cette première version, dite *exacte*, est que la première affectation $\bar{y}$ réalisable est nécessairement optimale ; le nombre d'itérations de la méthode est alors faible. La contre-partie de ce nombre réduit d'étapes est que l'on résout, à chaque itération, un problème NP-difficile ($[MP^r]$) de manière exacte. Un tel constat justifie alors l'étude d'une variante pour laquelle le problème maître serait résolu de manière appochée. Cette étude est présentée ci-après.

4.2.2 Résolution approchée

La seconde variante de la méthode de coupes repose sur la résolution approchée des problèmes maîtres. On ne recherche alors plus la solution optimale de chaque

problème maîte $[MP^r]$ mais une solution approchée. De ce fait, la première affectation $\bar{y}_r$ de roulement aux opérateurs qui s'avère réalisable pour $[P]$ n'est plus nécessairement optimale : il peut en effet exister une autre affectation réalisable de coût inférieur.

Supposons $\bar{y}_r$ une affectation de roulement aux opérateurs comme une solution réalisable de $[MP^r]$. Si le sous-problème $[SP(\bar{y}_r)]$ prouve que $\bar{y}_r$ est une solution réalisable de $[P]$, on ajoute une *coupe d'optimalité* à $[MP^r]$. Cette inégalité invalide les affectations de coût supérieur à celui de $\bar{y}_r$ (notons-le $c_{\bar{y}_r}$), elle peut être formalisée comme suit :

$$\sum_{o \in O} \sum_{\omega \in \Omega_o} \eta^o_\omega \cdot y^o_\omega < c_{\bar{y}_r} \tag{4.5}$$

Le processus global itère alors jusqu'à ce que $[MP^r]$ ne possède plus de solution réalisable ; la dernière affectation $\bar{y}_k$ $(k < r)$ réalisable étant optimale. L'algorithme 6 résume cette variante *approchée* de la méthode de génération de coupes.

Algorithme 6 Variante *approchée* de la méthode de génération de coupes

$UB \leftarrow \infty$, $r \leftarrow 0$
Répéter
  $\bar{y}_r \leftarrow$ solution réalisable de $[MP^r]$, de coût : $c_{\bar{y}_r}$
  Si $\bar{y}_r$ est définie **alors**
    $r \leftarrow r + 1$
    Si $f_{\bar{y}_r} < \sum_{j \in J} p_j$ **alors**
      ajouter la *coupe minimale de flot* (4.1) sur $G_{\bar{y}_r}$ à $[MP^r]$
    Sinon
      $UB \leftarrow c_{\bar{y}_r}$
      ajouter la *coupe d'optimalité* (4.5) à $[MP^r]$
    Fin Si
  Fin Si
Jusqu'à ($\bar{y}_r$ n'est pas définie)
Renvoyer UB

L'intérêt premier de cette seconde version, dite *approchée*, est que le temps de calcul induit par la recherche d'une solution aux problèmes maîtres $[MP^r]$ est réduit à chaque itération. En effet, trouver une solution réalisable à un problème ne prend pas plus de temps que trouver sa solution optimale. Une contrepartie de cette importante réduction de temps de calcul par itération est que la première affectation réalisable n'est plus nécessairement optimale. Il s'ensuit alors que le nombre d'itérations de cette méthode dépasse logiquement, en moyenne, le nombre d'itérations de la variante *exacte*.

Un second intérêt majeur à cette variante est qu'elle présente l'avantage de fournir, en cours de processus, des solutions réalisables de qualité. La légère contrepartie à ce nouvel atout est que cette version *approchée* nécessite une dernière itération devant prouver que le problème maître n'a plus aucune solution réalisable. Cette ultime étape est en effet nécessaire pour conclure à l'optimalité de la dernière affectation qui a été prouvée réalisable.

Dans nos expérimentations, deux techniques heuristiques de résolution de $[MP]$ ont été testées. La première consiste à utiliser un solveur de PLNE paramétré pour s'arrêter dès qu'une solution réalisable au problème maître a été trouvée. La seconde technique porte sur l'utilisation de l'heuristique de Feasibility Pump. Nous détaillons ci-après le cadre d'application de l'heuristique de Feasibility Pump et spécifions son utilisation dans le cadre de notre étude.

Heuristique de Feasibility Pump

Présentation dans le cadre général

L'heuristique de Feasibility Pump fût tout d'abord proposée par Fischetti et al. dans [FGL05], puis fût successivement développée par Bertacco et al. dans [BFL05], et Achterberg et Berthold dans [AB07]. Cette heuristique primale a pour but de trouver rapidement une solution réalisable de qualité à un Programme Linéaire Mixte générique de la forme $\min\{c^T x / Ax \geq b, (x_i) \in \mathbb{N} \quad \forall i \in I\}$. L'idée directrice de cette approche est de générer deux trajectoires convergentes de points partiellement réalisables. L'une satisfait les contraintes linéaires du modèle et l'autre les contraintes d'intégralité.

Un premier point (notons le x^*) vérifiant l'ensemble des contraintes linéaires est d'abord trouvé. Chaque composante $(x_i)^*$ de x^* (avec $i \in I$) est ensuite arrondie : un nouveau point $\tilde{x}$ vérifiant les contraintes d'intégralité est alors obtenu. Si l'une au moins des contraintes linéaires est violée par $\tilde{x}$, on recherche la solution réalisable (pour la relaxation continue du problème) la plus *proche*[3] de $\tilde{x}$. Cette nouvelle solution constitue alors le point continu x^* de l'itération suivante. Le processus itère jusqu'à ce que x^* et $\tilde{x}$ soient identiques. Afin d'éviter tout cycle dans la recherche, des phénomènes de perturbation sont mis en place (nous nous référons à [BFL05] pour une explication détaillée de ces phénomènes de perturbation).

La figure 4.4 illustre le fonctionnement de l'heuristique de Feasibility Pump. Sur cet exemple, on s'intéresse à un programme linéaire $[PL]$ à 2 variables et 3 contraintes. On cherche à déterminer une solution réalisable x à deux dimensions entières.

Une solution x_0^* réalisable pour la relaxation continue de $[PL]$ est tout d'abord déterminée. Aucune des dimensions de x_0^* n'étant entière, x_0^* n'est pas une solution réalisable de $[PL]$. Chaque composante de x_0^* est alors arrondie et un nouveau point

3. selon une fonction de distance prédéfinie

entier $\tilde{x}_1$ est ainsi obtenu. $\tilde{x}_1$ est en dehors du polyèdre associé à la relaxation continue de $[PL]$ (zone grisée sur l'illustration) : $\tilde{x}_1$ n'est du fait pas réalisable. On cherche donc la solution x_1^* comme étant le point vérifiant l'ensemble des contraintes linéaires de $[PL]$ le plus *proche* de $\tilde{x}_1$. À l'instar de x_0^*, x_1^* ne vérifie aucune des contraintes d'intégralité de $[PL]$. x_1^* n'est par conséquent pas réalisable, chacune de ces composantes est alors arrondie. On obtient ainsi le nouveau point *entier* $\tilde{x}_2$. $\tilde{x}_2$ s'avère être une solution à composantes entières et vérifiant l'ensemble des contraintes linéaires de $[PL]$: $\tilde{x}_2$ est dès lors une solution entière réalisable pour $[PL]$. L'heuristique de Feasibility Pump est en conséquence arrivée à son but et peut être stoppée.

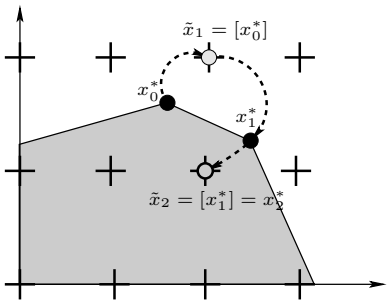

FIGURE 4.4 – Exemple d'application de l'heurisitique de Feasibility Pump

Application à notre étude

Le problème maître $[MP]$ de la technique de coupes décrite dans ce chapitre est un programme linéaire en variables binaires, sa résolution constitue donc un cas particulier du champ d'application de l'heuristique de Feasibility Pump. Nous détaillons dans les points suivants les caractéristiques utilisées dans nos expérimentations.

Solution continue initiale : la solution optimale de la relaxation continue du problème maître est utilisée comme solution continue initiale.

L'intérêt est ici de converger plus probablement vers une solution réalisable globale de qualité. L'heuristique de Feasibility Pump étudiant les solutions par voisinage *proche*, il est en effet raisonnable de penser que débuter avec une solution continue de qualité peut converger plus sûrement vers une solution entière réalisable de qualité. On suppose bien sûr ici (ce qui est vrai en pratique) que la relaxation continue de chaque problème maître $[MP]$ constitue une borne inférieure de qualité.

Règle d'arrondi, passage d'une solution continue y^* à une solution entière $\tilde{y}$: on applique à chaque composante de y^* une technique d'arrondi classique, à savoir :

$$\forall o \in O \quad \forall \omega \in \Omega_o \quad \tilde{y}_\omega^o = \lfloor (y_\omega^o)^* + 0.5 \rfloor$$

Fonction de distance, passage d'une solution entière $\tilde{y}$ à une solution continue y^* : nous considérons la même distance Δ que celle proposée dans leurs travaux par Fischetti et al. [FGL05], à savoir : la distance de *Manhattan* (ou 1-distance). Δ se définit alors comme suit :

$$\Delta(y, \tilde{y}) = \sum_{o \in O} \sum_{\omega \in \Omega_o} |y_\omega^o - \tilde{y}_\omega^o|$$

Dans notre cas, chaque composante de $\tilde{y}$ est (du fait du domaine de définition $\{0, 1\}$ des variables y_ω^o) nécessairement binaire. Δ se réécrit par conséquent sous la forme linéaire :

$$\Delta(y, \tilde{y}) = \sum_{o \in O} \sum_{\substack{\omega \in \Omega_o | \\ \tilde{y}_\omega^o = 0}} y_\omega^o + \sum_{o \in O} \sum_{\substack{\omega \in \Omega_o | \\ \tilde{y}_\omega^o = 1}} (1 - y_\omega^o)$$

Pour passer d'une solution $\tilde{y}$ vérifiant les contraintes d'intégralité à une solution y^* vérifiant les contraintes linéaires, on résout alors (via un solveur de programmation linéaire) le programme linéaire continu $[FPMP_{\tilde{y}}]$ [4] suivant :

$$\begin{aligned}
[FPMP_{\tilde{y}}] : \min \quad & \sum_{o \in O} \sum_{\substack{\omega \in \Omega_o | \\ \tilde{y}_\omega^o = 0}} y_\omega^o + \sum_{o \in O} \sum_{\substack{\omega \in \Omega_o | \\ \tilde{y}_\omega^o = 1}} (1 - y_\omega^o) & \\
\forall o \in O \qquad & \sum_{\omega \in \Omega_o} y_\omega^o & = 1 \\
& \mathit{Coupes} & \\
\forall o \in O, \forall \omega \in \Omega_o \qquad & y_\omega^o \in [0, 1] &
\end{aligned}$$

Coupes désignant l'ensemble des coupes ajoutées itérativement au problème maître $[MP]$ de la technique de coupes.

L'efficacité présumée de l'heuristique de Feasibility Pump tient dans le postulat que chaque programme linéaire $[FPMP_{\tilde{y}}]$ se résout rapidement. Sous cette hypothèse, le processus itératif de cette heuristique est composé d'étapes nécessitant toutes un temps de calcul réduit. La méthode converge alors vite vers une solution entière réalisable.

4. $FPMP$ étant acronyme de Feasibility Pump Master Problem

Chapitre 5

Inégalités initiales valides

La vitesse de convergence des méthodes de coupes décrites ci-dessus est très fortement tributaire de la qualité des coupes générées en cours de résolution. Une amélioration directe à ces techniques, mais également à d'autres méthodes potentielles, consiste alors à fournir des coupes initiales valides en entrée de processus. Nous présentons dans ce chapitre certaines investigations quant à l'initialisation d'un ensemble d'inégalités valides pour $[P]$.

5.1 Raisonnement énergétique

Le raisonnement utilisé pour définir ces inégalités est lié au concept d'*énergie*. Ce concept a été très largement utilisé avec succès (notamment par Lopez et al. dans [LEE92], et Baptiste et al. dans [BLPN99]) dans des problèmes d'ordonnancement. Dans cette idée, les compétences sont considérées comme des ressources consommables, les opérateurs comme des fournisseurs de ressources et les jobs comme des activités consommatrices. Ce raisonnement prend tout son intérêt sur des périodes de temps où il est possible d'établir, pour un sous-ensemble de jobs, une *consommation obligatoire* en compétences. Sur de telles périodes, la disponibilité en ressources concernées doit être suffisamment élevée pour permettre la consommation.

Considérons une période de temps $\delta = [\delta_b, \delta_e] \in \Delta$ avec $\Delta \subseteq \mathcal{P}(H)$, un sous-ensemble de compétences $\mathcal{C} \subseteq C$ et un sous-ensemble de jobs $\mathcal{J} \subseteq J$.

Par la suite, nous utiliserons la notation $(a)^+ = \max(a, 0) \quad \forall a \in \mathbb{R}$.

Consommation obligatoire d'un job j durant δ La consommation obligatoire de j durant δ (notons-la $u_{j\delta}$) correspond à la différence entre p_j et le nombre d'unités de j qu'il est possible d'exécuter en dehors de δ :

$$u_{j\delta} = \big(p_j - (\delta_b - r_j)^+ - (d_j - \delta_e)^+\big)^+$$

Consommation minimale en compétence c durant δ La consommation minimale en compétence c durant δ (notons-la $U_{c\delta}$) correspond à l'ensemble des *consommations obligatoires* sur δ des jobs requérant c :

$$U_{c\delta} = \sum_{\substack{j \in J | \\ c_j = c}} u_{j\delta}$$

Capacité des ressources disponibles $\mathcal{C}$ durant δ Durant δ, chaque opérateur peut travailler autant d'instants qu'il est *présent* selon le roulement qui lui est affecté. Toutefois, il ne peut utiliser qu'une compétence à la fois. Par conséquent, dès lors qu'un opérateur o est *présent* et qu'il maîtrise une compétence de $\mathcal{C}$, une et une seule unité de compétence de o est disponible par instant $t \in \delta$. La capacité de ressources disponibles $\mathcal{C}$ durant δ (notons la $capa_{\delta\mathcal{C}}$) peut donc être formulée comme suit :

$$capa_{\delta\mathcal{C}} = \sum_{t \in \delta} \sum_{\substack{o \in O | \\ \{C_o \cap \mathcal{C} \neq \emptyset\}}} \sum_{\omega \in \Omega_o} \sigma_\omega^t \cdot y_\omega^o$$

Contrainte *énergétique* Toute solution réalisable de $[P]$ vérifie par conséquent nécessairement la contrainte *énergétique* suivante :

$$capa_{\delta\mathcal{C}} \geq \sum_{c \in \mathcal{C}} U_{c\delta}$$

Pour Exemple 1, une contrainte énergétique peut être générée durant $\delta = [1, 7]$ avec $\mathcal{C} = \{c_1\}$ (cf. figure 5.1).

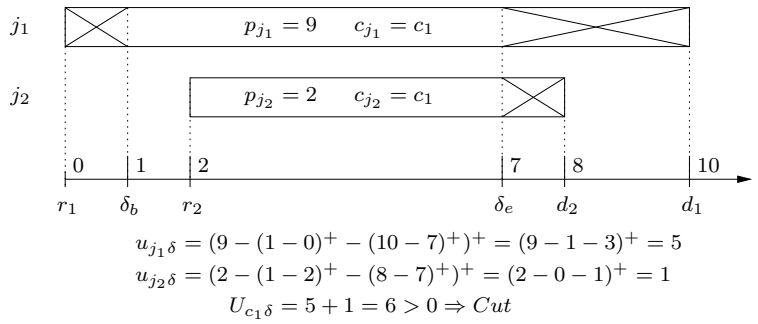

FIGURE 5.1 – Contrainte énergétique pour $\delta = [1, 7]$ et $\mathcal{C} = \{c_1\}$

Sur la figure 5.1, on cherche à savoir si une coupe peut être générée sur $\delta = [1, 7]$ avec $\mathcal{C} = \{c_1\}$. On considère alors les deux jobs j_1 et j_2 nécessitant tous deux la compétence $c_1 \in \mathcal{C}$. j_1 a pour domaine de définition $[0, 10]$ et est de durée 9, sa consommation obligatoire $u_{1\delta}$ sur δ est donc 5. De la même manière, on trouve que la consommation obligatoire $u_{2\delta}$ du job j_2 sur δ est 1. Par conséquent, la consommation minimale $U_{1\delta}$ en compétence c_1 durant δ est 6. 6 étant strictement positif, une coupe énergétique informative peut être définie.

5.2 Initialisation du pool de contraintes initiales

Il existe un très grand nombre de périodes de temps δ (non nécessairement connexes), et donc de contraintes énergétiques possibles. Toutes les générer s'avère trop coûteux en temps de calcul. Nous avons donc mis en place des règles gloutonnes quant au choix des périodes de temps et des sous-ensembles de compétences pour lesquels nous générons des contraintes énergétiques.

Nous définissons tout d'abord $\tau_\delta^c = \dfrac{U_{c\delta}}{\delta_e - \delta_b}$ comme la *consommation proportionnelle en c obligatoire durant* δ. Cet indicateur nous permet de sélectionner des périodes de temps vraisemblablement plus contraignantes. Si on considère deux périodes $\delta \in \Delta$ et $\delta' \in \Delta$ telles que $\tau_\delta^c > \tau_{\delta'}^c$, on supposera que la coupe générée selon c sur δ est plus significative que celle générée sur δ'.

L'idée sous-jacente de la sélection de périodes de temps retenue dans nos expérimentations repose sur l'intérêt tout particulier consacré à l'ensemble Υ des bornes υ des intervalles de présence de chaque roulement. Ces instants sont en effet les seuls où le nombre d'opérateurs présents (et donc de compétences disponibles) peut varier.

Soient $[\![\upsilon_i, \upsilon_{i+1}]\!]$ et $[\![\upsilon_{j-1}, \upsilon_j]\!]$ deux périodes de temps telles que $(i,j) \in |\Upsilon|^2$ et $\upsilon_i < \upsilon_j$. On génère, pour chaque paire $(i,j) \in |\Upsilon|^2$, une contrainte énergétique pour la période de temps δ définie comme suit :

$$\delta = \underset{\delta' = [\delta'_b, \delta'_e]}{\operatorname{argmax}} \tau_{\delta'}^{\mathcal{C}}$$

avec δ'_b et δ'_e deux bornes d'intervalles de temps I_k ($k \in K = [\![1, k_{max}]\!]$) appartenant respectivement à $[\![\upsilon_i, \upsilon_{i+1}]\!]$ et $[\![\upsilon_{j-1}, \upsilon_j]\!]$.

Pour Exemple 1, on s'intéressera par exemple à déterminer une coupe énergétique pour les périodes de temps $[\![\upsilon_i, \upsilon_{i+1}]\!] = [\![0, 8]\!]$ et $[\![\upsilon_{j-1}, \upsilon_j]\!] = [\![10, 16]\!]$. On cherchera alors la coupe la plus significative au sens de sa consommation proportionnelle obligatoire τ parmi celles que l'on peut générer respectivement sur $[\![0, 10]\!]$, $[\![0, 16]\!]$, $[\![2, 10]\!]$, $[\![2, 16]\!]$, $[\![8, 10]\!]$ et $[\![8, 16]\!]$.

Dans nos expérimentations, nous définissons une contrainte énergétique pour chaque paire $(i,j) \in |\Upsilon|^2$ et chaque sous-ensemble de compétences $\mathcal{C} \in \{\Theta \cup \mathcal{S}\}$ avec Θ l'ensemble des profils de compétences et $\mathcal{S}$ l'ensemble des singletons (une unique compétence).

5.3 *Affinement* des contraintes

Les calculs des coefficients et du second membre de chaque contrainte énergétique étant indépendants, il est possible d'obtenir des coupes dont les coefficients soient

nettement supérieurs au second membre. Les variables de décision étant toutes binaires et les coupes énergétiques étant des inégalités de type *supérieur ou égal*, un tel état de choses est inutile et peut parasiter la recherche de solution (notamment pour un solveur de programmation linéaire). Des techniques de réduction de coefficients et de second membre ont alors été mis en place afin d'*affiner* -ou *resserrer*- les contraintes initiales.

Considérons une contrainte initiale i portant sur un ensemble de compétences $\mathcal{C}$ durant δ, i est de la forme :

$$\sum_{o \in O} \sum_{\omega \in \Omega_o} \alpha^i_{o\omega} \cdot y^o_\omega \geq b_i$$

où $\alpha^i_{o\omega} = \sum_{t \in \delta} \sigma^t_\omega$ si $C_o \cap \mathcal{C} \neq \emptyset$, 0 sinon ; et $b_i = \sum_{c \in \mathcal{C}} U_{c\delta}$.

Chaque coefficient $\alpha^i_{o\omega}$ étant un entier naturel ($o \in O$ et $\omega \in \Omega_o$), on peut calculer le Plus Grand Commun Diviseur $\bar{\alpha}^i_{o\omega}$ de l'ensemble des coefficients $\alpha^i_{o\omega}$. On pose ainsi $\bar{\alpha}^i_{o\omega} = \text{PGCD}_{(o \in O) \wedge (\omega \in \Omega_o)} \alpha^i_{o\omega}$. Chaque coefficient $\alpha^i_{o\omega}$ est alors divisé par $\bar{\alpha}^i_{o\omega}$, $\alpha^i_{o\omega}$ restant entier.

Le second membre b_i est ensuite divisé de la même manière par $\bar{\alpha}^i_{o\omega}$. $\bar{\alpha}^i_{o\omega}$ étant strictement positif, le sens de la contrainte i ne change pas. Étant admis que chaque coefficient $\alpha^i_{o\omega}$ de i est entier et que chaque variable y^o_ω appartient à $\{0,1\}$, la valeur de la coupe est nécessairement entière. Il est alors valide d'arrondir à l'entier supérieur le second membre b_i que l'on vient de déterminer.

Les coefficients $\alpha^i_{o\omega}$ sont enfin réduits à hauteur de b_i si nécessaire. On pose alors $\alpha^i_{o\omega} = \min(\alpha^i_{o\omega}, b_i)\ \forall o \in O\ \forall \omega \in \Omega_o$.

Pour illustrer cette opération d'*affinement*, on étudie la contrainte énergétique *brute* exemple suivante :

$$10 \cdot y^1_1 + 40 \cdot y^2_1 \geq 3 \tag{5.1}$$

Les étapes d'*affinement* réalisées sont alors, dans l'ordre :

1. $PGCD(10, 40) = 10$, $(5.1) \Leftrightarrow y^1_1 + 4 \cdot y^2_1 \geq 0.3$
2. $\lceil 0.3 \rceil = 1$, $(5.1) \Rightarrow y^1_1 + 4 \cdot y^2_1 \geq 1$
3. $(\min(1,1) = 1) \wedge (\min(4,1) = 1)$, $(5.1) \Rightarrow y^1_1 + y^2_1 \geq 1$

L'étape 2 de ce processus d'*affinement* a une spécificité toute particulière. Lors de cette phase de calcul, on *resserre* chaque contrainte énergétique dans le sens où l'aire du polytope des solutions réalisables est réduit sans qu'aucune solution entière admissible ne soit éliminée. La relaxation continue de nos modèles est ainsi renforcée.

5.4 Suppression des contraintes dominées

Le processus de génération de coupes défini ci-dessus considérant les périodes de temps indépendamment les unes des autres, il n'est pas impossible (et même plus que probable) que certaines coupes générées soient *dominées*.

Définition 1
Soient deux contraintes énergétiques $i : \sum_{o \in O} \sum_{\omega \in \Omega_o} \alpha^i_{o\omega} \cdot y^o_\omega \geq b_i$ *et* $i' : \sum_{o \in O} \sum_{\omega \in \Omega_o} \alpha^{i'}_{o\omega} \cdot y^o_\omega \geq b_{i'}$ *($i' \neq i$), avec* $b_i > 0$ *et* $b_{i'} > 0$.

On dira que i domine i' si et seulement si

$$\forall o \in O, \forall \omega \in \Omega_o \quad \alpha^i_{o\omega} \cdot \frac{b_{i'}}{b_i} \leq \alpha^{i'}_{o\omega}$$

Dans cette définition, il est entendu que seules les contraintes énergétiques i à second membre (b_i) strictement positif sont générées.

Dans nos expérimentations, nous éliminons chaque contrainte dominée de l'ensemble des contraintes initiales.

5.5 Sélection de contraintes

Le nombre de coupes non dominées générées s'avére expérimentalement très élevé. Cet état de fait est alors contre-productif et nuit à la vitesse de convergence des méthodes utilisant les coupes initiales. Le gain de temps de résolution induit par l'information supplémentaire donnée par de telles coupes est en effet trop négativement contre-balancé par la perte de temps subie par les techniques pour traiter tant d'informations. Il convient alors de trouver un juste équilibre, et donc de sélectionner un sous-ensemble des coupes initiales les plus *significatives*.

Le mode de sélection que nous avons défini consiste tout d'abord à résoudre la relaxation continue de $[P]$ en y incluant l'ensemble des contraintes non dominées. On sélectionne ensuite les contraintes dont le rapport entre le *gap* à l'optimum continu et le second membre est inférieur à 12% (valeur empirique fixée après expérimentation). On émet ainsi ici le posutlat que les contraintes ayant de faibles *gaps* à l'optimum continu sont les plus *significatives*.

Les contraintes non sélectionnées (mais non dominées) sont toutefois conservées dans les méthodes de coupes (cf. chapitres 3 et 4). En effet, à chaque itération de tels processus, on peut ajouter aux programmes maîtres chaque contrainte (non sélectionnée) violée par l'affectation en cours de roulement aux opérateurs.

5.6 Intérêt des coupes initiales

Expérimentalement, les coupes initiales s'avèrent d'une grande utilité car elles fournissent, à un moindre coût de temps de calcul, des informations autorisant l'invalidation d'un grand nombre d'affectations irréalisables. De plus, de par leur nature, elles mettent rapidement en exergue de fortes obligations d'affectation sur des périodes précises de l'horizon. Elles diffèrent alors sigulièrement des coupes de réalisabilité générées par les méthodes de coupes (cf. chapitre 3 et 4) car ces dernières ont une vision plus générale du problème. Par conséquent, l'information retirée par l'usage des inégalités *initiales* peut être longue à retrouver par les méthodes de coupes. Ainsi, ces différents types d'inégalités sont complémentaires et leur utilisation commune est réellement bénéfique.

Nous remarquons de plus que les coupes de réalisabilité de la méthode de coupes décrite au chapitre 4, et les inégalités initiales ont des structures similaires. Le problème maître de cette technique conserve donc la même nature, à savoir un problème de *sac à dos multi-choix multi-dimensionnel*. Les techniques dédiées à cette classe de problème restent donc applicables lorsque les coupes initiales sont ajoutées en pré-process.

Chapitre 6

Expérimentations numériques

Les diverses méthodes décrites dans cette première partie ont été implémentées puis testées expérimentalement sur des instances générées. Nous exposons et discutons les résultats numériques de chaque technique dans ce chapitre.

Nous définissons dans un premier temps le matériel utilisé, puis détaillons les caractéristiques de notre générateur de données. Nous regroupons ensuite les méthodes selon leur catégorie de sortie : borne inférieure, borne supérieure, optimum. Les expérimentations numériques de chaque méthode sont alors présentées par catégorie.

6.1 Matériel

Les techniques de résolution ont été implémentées en Java et testées sur un PC (Intel Pentium D 930, 3 GHz, 2 GB RAM) dont le système d'exploitation est MS Windows XP.

Le solveur de programmation linéaire (MIP) utilisé est ILOG CPLEX 9.1. Les paramètres du solveur sont ceux par défaut, à l'exception du paramètre *MipEmphasis* qui gère le guidage de recherche pour les MIP. Ce paramètre est fixé à *Feasibility* pour chaque méthode ayant recours au solveur.

6.2 Instances de test

Notre jeu de données est composé de données aléatoires, les jobs sont générés en premier. Leur date de démarrage au plus tôt, leur durée et leur marge (taille du domaine de définition) sont respectivement distribuées selon une loi uniforme et deux lois binomiales. Une durée maximale p_{max} ainsi qu'une marge maximale $marge_{max}$ sont imposées identiquement à chaque job.

Les roulements sont ensuite générés. Afin de coller au mieux à la réalité des usines de production, les roulements de nos instances respectent les règles de *travail en trois-huit* (ceci constitue un choix qui ne remet pas en cause l'universalité de nos méthodes ; ces dernières étant tout à fait applicables à d'autres modes de

fonctionnement manufacturier). Le quart d'heure est l'unité de temps retenue pour la génération des roulements. L'attribution d'un ensemble de roulements éligibles et d'un ensemble de compétences maîtrisées est réalisée de manière aléatoire pour chaque opérateur. Enfin, un coût est calculé pour chaque paire opérateur - roulement éligible. Ce coût dépend des horaires du roulement et des spécificités de l'opérateur (expérience, qualification, préférences).

Le tableau 6.1 synthétise les paramètres utilisés pour générer nos 270 instances de test. Notons que 3 instances sont générées pour chaque jeu de paramètres.

Paramètre	min	max	pas
m	15	25	10
n	$4 \cdot m$	$6 \cdot m$	m
$marge_{max}$	30	90	30
$\|C\|$	1	5	1
p_{max}	30		
$\|H\|$	480		

TABLE 6.1 – Paramètres de la génération des données

Sauf indication contraire, la limite de temps CPU imposée pour chaque technique est de 5 minutes.

6.3 Formalisation

Le tableau 6.2 regroupe les performances du solveur MIP appliqué à la formalisation indexée sur le temps $[Q]$ et à la formalisation compacte $[P]$ du problème. Les colonnes *Succès* et *Temps(S)* indiquent respectivement le pourcentage d'instances résolues et le temps moyen de résolution (uniquement pour les instances résolues optimalement) de chaque méthode.

		(MIP) :[Q]		(MIP) :[P]	
m	$\|C\|$	Succès	Temps(S)	Succès	Temps(S)
15	1-2	27.8%	72.8s	**83.3%**	**31.0s**
	3-5	35.8%	110.5s	**86.4%**	**44.1s**
$(m = 15)$		32.6%	97.7s	**85.2%**	**39.0s**
25	1-2	14.8%	151.6s	**46.3%**	**31.7s**
	3-5	13.6%	151.8s	**38.3%**	**56.4s**
$(m = 25)$		14.1%	151.7s	**41.5%**	**45.4s**
$(Total)$		23.3%	114.0s	**63.3%**	**41.1s**

TABLE 6.2 – Performance des formalisations

La formalisation compacte $[P]$ permet de résoudre environ 3 fois plus d'instances que la formalisation indexée sur le temps $[Q]$, et dans un temps nettement inférieur. Sur les 63 instances résolues optimalement via les deux formalisations $[Q]$ et $[P]$, le solveur est 23 fois plus rapide lorsqu'il est appliqué à la formalisation compacte $[P]$.

6.4 Bornes inférieures

Afin d'évaluer les performances des deux bornes inférieures lagrangiennes *totale* ($LagTot$) et *approchée* ($LagPart$) décrites au chapitre 2, nous comparons (dans le tableau 6.3) leurs résultats à ceux fournis par un solveur de programmation linéaire sur la relaxation continue (LP) associée à la formalisation compacte $[P]$.

		(LagTot)		(LagPart)		(LP)
m	$\|C\|$	Temps	(LagTot) / (LP)	Temps	(LagPart) / (LP)	Temps
15	1-2	0.3s	95.9%	1.7s	**97.1%**	**0.1s**
	3-5	1.7s	94.9%	5.2s	**95.9%**	**0.2s**
$(m = 15)$		1.1s	95.3%	3.8s	**96.4%**	**0.2s**
25	1-2	0.6s	96.7%	4.0s	**97.0%**	**0.2s**
	3-5	4.1s	95.5%	14.4s	**95.7%**	**0.9s**
$(m = 25)$		2.7s	96.0%	10.2s	**96.2%**	**0.6s**
$(Total)$		1.9s	95.6%	7.0s	**96.3%**	**0.4s**

TABLE 6.3 – Performance des bornes inférieures

La décomposition lagrangienne *totale* ($LagTot$) nécessite en moyenne 5 fois plus de temps de calcul que la relaxation continue (LP) pour atteindre une borne de qualité semblable (ratio $(LagTot)/(LP) \geq 95.0\%$). À temps de calcul équivalent, le ratio $(LagTot)/(LP)$ est de l'ordre de 85%. Ceci est essentiellement dû à la trop forte dénaturation du problème initial pour le calcul de cette décomposition *totale*. En effet, un très grand nombre de contraintes sont ici dualisées. La difficulté théorique du problème est alors transposée en grande partie à la méthode de sous-gradient qui peine à converger rapidement vers une borne de qualité.

Pour sa part, la décomposition lagrangienne *partielle* ($LagPart$) nécessite en moyenne 10 fois plus de temps de calcul que la relaxation continue pour atteindre une borne de qualité semblable. Le ratio $(LagPart)/(LP)$ est ici de l'ordre de 80% à temps de calcul équivalent. En pratique, on s'aperçoit que cette décomposition *partielle* converge toutefois plus rapidement que la décomposition *totale* ($LagTot$) en nombre d'itérations de la méthode de sous-gradient. Malheureusement, le temps de calcul de la fonction duale en chaque itération est ici rédhibitoire et ce, essentiellement en raison du calcul coûteux d'un flot maximal de coût minimal induit par l'un des deux sous-problèmes lagrangiens.

6.5 Bornes supérieures

Nous proposons désormais d'évaluer les bornes supérieures obtenues en appliquant la variante *approchée* de la méthode de génération de coupes décrite en 4.2.2. On compare pour cela les résultats obtenus pour les trois approches suivantes :

- (MIP) : solveur MIP appliqué à la formalisation compacte $[P]$.
- $(Coupe : [MP] \hookrightarrow MIP)$: variante *approchée* de la méthode de génération de coupes, chaque problème maître de la méthode étant résolu via le solveur MIP paramétré pour s'arrêter dès qu'une solution entière a été trouvée (paramètre *IntSolLim* fixé à 1).
- $(Coupe : [MP] \hookrightarrow Feasibility\ Pump)$: variante *approchée* de la méthode de génération de coupes, chaque problème maître de la méthode étant résolu via l'heuristique de Feasibility Pump.

Le tableau 6.4 présente une synthèse des résultats pour les trois techniques. Le ratio UB^*/UB est utilisé comme indicateur de performance pour chaque méthode, UB^* et UB représentant respectivement la meilleure borne supérieure connue pour l'instance (l'optimum dans 90.4 % des cas) et la borne supérieure trouvée par la technique concernée. Les colonnes UB^*/UB et σ indiquent respectivement la moyenne et l'écart-type des ratios UB^*/UB de chaque catégorie d'instances.

Notons qu'un même sous-ensemble de coupes initiales (40.6 en moyenne) est ajouté en pré-process de chaque méthode. On verra dans la section suivante que ces coupes initiales améliorent significativement les performances de l'ensemble de nos méthodes.

On remarque tout d'abord que $(Coupe : [MP] \hookrightarrow MIP)$ et $(Coupe : [MP] \hookrightarrow Feasibility\ Pump)$ convergent toutes deux très rapidement vers des solutions approchées de très bonne qualité. Pour atteindre un ratio moyen UB^*/UB de l'ordre de 95.5%, le solveur MIP nécessite en effet 2 fois plus de temps de calcul que $(Coupe : [MP] \hookrightarrow MIP)$, et 10 fois plus que $(Coupe : [MP] \hookrightarrow Feasibility\ Pump)$. La variante *approchée* de la méthode dédiée de génération de coupes s'avère donc être particulièrement adaptée à notre problématique.

On observe également que $(Coupe : [MP] \hookrightarrow Feasibility\ Pump)$ est, quelque soit le temps limite de calcul imposé, plus efficace en termes de moyenne et d'écart-type du ratio UB^*/UB que $(Coupe : [MP] \hookrightarrow MIP)$. L'usage de l'heuristique de *Feasibility Pump* est par conséquent tout indiqué pour la résolution de chaque problème maître (sac à dos multi-Choix multi-dimensionnel) de la technique de coupes.

Temps limite	(*MIP*) avec coupes initiales		(Coupe) avec coupes initiales			
			([*MP*] $\hookrightarrow$ *MIP*)		([*MP*] $\hookrightarrow$ *Feasibility Pump*)	
	UB^*/UB	σ	UB^*/UB	σ	UB^*/UB	σ
1s	77.6 %	21.0%	87.2%	19.0%	**95.4%**	**10.4%**
5s	92.2 %	15.5%	95.6%	11.7%	**99.0%**	**3.5%**
10s	95.6 %	13.2%	98.2%	6.8%	**99.5%**	**1.1%**
30s	99.1 %	4.2%	99.1%	4.6%	**99.8%**	**0.5%**
60s	99.6 %	0.6%	99.5%	3.2%	**99.9%**	**0.4%**
300s	99.9 %	**0.2%**	99.9%	0.5%	**100.0%**	**0.2%**

TABLE 6.4 – Performance des bornes supérieures

6.6 Méthodes exactes

Nous nous intéressons désormais aux résultats des différentes méthodes exactes proposées dans ce manuscrit. Nous reportons dans un premier temps les performances de chaque technique selon que les coupes initiales décrites dans le chapitre 5 sont utilisées ou non. Une synthèse comparative des versions les plus efficaces de chaque méthode est enuite proposée.

6.6.1 Coupes initiales

Le tableau 6.5 présente un descriptif des ensembles de coupes initiales (cf. chapitre 5) définis pour chaque instance. Dans nos expérimentations, chaque instance de test se voit affecter un unique ensemble de coupes initiales ; les techniques de résolution utilisant alors toutes ce même ensemble.

La colonne *# coupes initiales sélectionnées* donne le nombre moyen de coupes initiales sélectionnées pour chaque jeu d'instances. Ces coupes sont considérées comme les plus significatives, elles sont ainsi ajoutées au modèle de chaque technique dès son initialisation. La colonne *# autres coupes initiales* présente le nombre moyen de coupes initiales valides générées mais non sélectionnées. Lors de la sélection intensive des coupes initiales, ces coupes se sont révelées non significatives. Elles ne sont alors pas incluses dans les modèles des techniques de résolution dès le début de la résolution mais peuvent, selon la technique utilisée, être ajoutées en cours de processus. La colonne *Temps* expose enfin le temps moyen total consacré au calcul et à la sélection des coupes initiales.

m	$\|C\|$	# coupes initiales sélectionnées	# autres coupes initiales	Temps
15	1-2	25.8	101.4	0.2s
	3-5	27.5	439.8	0.9s
$(m = 15)$		26.8	304.4	0.6s
25	1-2	30.5	117.9	0.4s
	3-5	30.9	731.1	2.7s
$(m = 25)$		30.8	485.8	1.8s
$(Total)$		28.8	395.1	1.2s

Table 6.5 – Statistiques sur les coupes initiales

Nous observons tout d'abord que le temps de calcul et de sélection des coupes initiales est très faible (de l'ordre de la seconde). Le temps de calcul des coupes initiales n'est donc absolument pas un frein à leur utilisation.

Nous constatons ensuite que le nombre total de coupes initiales (cumul des *# coupes initiales sélectionnées* et des *# autres coupes initiales*) croît logiquement avec

la taille des instances. Cependant, le nombre de *coupes initiales sélectionnées* reste à peu près constant. On peut alors en conclure que le nombre de coupes initiales *presque serrées* à l'optimum relaxé du problème complet serait, de manière assez inattendue, indépendant de la taille du problème.

Nous remarquons enfin qu'en moyenne seulement 7% des coupes initiales sont sélectionnées. Ce mode de sélection résultant d'expérimentations intensives, il convient alors de conclure que l'usage de la génération de coupes initiales se doit de choisir de manière drastique les inégalités qu'elle relève. Sans cela, le gain apporté est contrebalancé de manière trop forte par la perte en temps de calcul induit par la gestion de tant d'informations supplémentaires.

6.6.2 Résultats individuels

Dans cette partie de l'exposé, nous présentons les résultats individuels des techniques de résolution. Pour chaque méthode exacte, on exprime les résultats obtenus selon que les coupes initiales déterminées en préprocess sont, oui ou non, utilisées.

Dans chacun des tableaux suivants, la colonne *Succès* présente le pourcentage d'instances résolues dans le temps imparti (5 minutes par instance). La colonne *Temps(S)* synthétise pour sa part le temps moyen de résolution des instances résolues optimalement. Une dernière colonne, $UB^*/z(E)$ ou $z/UB^*(E)$ selon la technique de résolution, donne enfin la déviation moyenne à la meilleure borne supérieure connue pour les instances non résolues optimalement [5] ; UB^* et z représentant respectivement la meilleure borne supérieure connue de l'instance (l'optimum dans 90.4 % des cas) et la borne (inférieure ou supérieure) trouvée par la technique concernée.

6.6.3 MIP

Le tableau 6.6 présente les résultats du solveur MIP appliqué à la formalisation compacte $[P]$ du problème.

Le solveur MIP affiche en moyenne un taux de succès de 74.1% avec les coupes initiales, contre 63.3% sans. On peut de plus observer que le temps de résolution est nettement amélioré lorsque les coupes initiales sont utilisées. L'usage de ces inégalités est donc particulièrement efficace pour cette première approche exacte.

6.6.4 Benders

Le tableau 6.7 synthétise les résultats de la décomposition de Benders présentée au chapitre 3.

L'usage des coupes initiales renforce là encore nettement les performances de la technique de résolution, ici basée sur une décomposition de Benders. Le taux moyen de succès passe en effet de 46.3% à 73.3% lorsque ces dernières sont utilisées.

5. E étant acronyme d'Échec

		(MIP)			(MIP) avec coupes initiales		
m	$\|C\|$	Succès	Temps(S)	z/UB^*(E)	Succès	Temps(S)	z/UB^*(E)
15	1-2	83.3%	31.0s	100.0%	**96.3%**	9.5s	100.0%
	3-5	86.4%	44.1s	99.9%	**97.5%**	35.0s	99.8%
$(m = 15)$		85.2%	39.0s	99.9%	**97.0%**	24.9s	99.9%
25	1-2	46.3%	31.7s	99.9%	**64.8%**	40.1s	99.9%
	3-5	38.3%	56.4s	99.6%	**42.0%**	40.1s	99.7%
$(m = 25)$		41.5%	45.4s	99.7%	**51.1%**	40.1s	99.7%
$(Total)$		63.3%	41.1s	99.8%	**74.1%**	30.1s	99.7%

Table 6.6 – Solveur MIP

		(Benders)			(Benders) avec coupes initiales		
m	$\|C\|$	Succès	Temps(S)	z/UB^*(E)	Succès	Temps(S)	z/UB^*(E)
15	1-2	83.3%	27.1s	97.1%	**100.0%**	**9.0s**	X
	3-5	30.9%	50.2s	96.0%	**86.4%**	**34.2s**	99.0%
$(m = 15)$		51.9%	35.4s	96.2%	**91.9%**	**23.2s**	99.0%
25	1-2	53.7%	22.4s	96.1%	**70.4%**	**15.8s**	99.2%
	3-5	32.1%	66.8s	96.9%	**44.4%**	**28.2s**	97.9%
$(m = 25)$		40.7%	43.4s	96.7%	**54.8%**	**21.9s**	98.2%
$(Total)$		46.3%	38.9s	96.4%	**73.3%**	**22.7s**	98.4%

Table 6.7 – Décomposition de Benders

En comparaison avec le solveur MIP, on s'aperçoit également que la décomposition de Benders avec coupes initiales affichent un taux de réussite semblable au solveur mais pour un temps de calcul moyen inférieur d'environ 25%.

6.6.5 Génération de coupes

Le tableau générique 6.8 exprime les résultats des différentes variantes de la génération de coupes présentée au chapitre 4. Le premier tableau est représentatif de la variante *exacte* ($Coupe$) de la méthode. Les deuxième et troisième tableaux montrent pour leur part les résultats de la variante *approchée* de la technique, selon que l'on utilise le solveur MIP ($Coupe : [MP] \hookrightarrow MIP$) ou l'heuristique de Feasiblity Pump ($Coupe : [MP] \hookrightarrow F.P.$) pour résoudre chaque problème maître de cette méthode de décomposition.

Une première observation de ces trois tableaux permet tout d'abord d'affirmer l'hypothèse d'efficacité probante des coupes initiales. Quelque soit la variante de la méthode de coupes, les résultats se trouvent en effet très nettement améliorés dès lors que les coupes initiales sont utilisées.

		(*Coupe*)			(*Coupe*) avec coupes initiales		
m	$\|C\|$	Succès	Temps(S)	z/UB^*(E)	Succès	Temps(S)	z/UB^*(E)
15	1-2	85.2%	33.2s	97.3%	**100.0%**	5.5s	X
	3-5	53.1%	88.1s	94.3%	**80.2%**	31.8s	98.4%
($m = 15$)		65.9%	59.8s	94.9%	**88.1%**	19.9s	98.4%
25	1-2	55.6%	17.5s	96.0%	**77.8%**	21.1s	90.5%
	3-5	40.7%	53.9s	96.3%	**49.4%**	48.2s	90.9%
($m = 25$)		46.7%	36.6s	96.2%	**60.7%**	34.3s	90.8%
(*Total*)		56.3%	50.2s	95.7%	**74.4%**	25.7s	92.5%

		(*Coupe* : $[MP] \hookrightarrow MIP$)			(*Coupe* : $[MP] \hookrightarrow MIP$) avec coupes initiales		
m	$\|C\|$	Succès	Temps(S)	UB^*/z(E)	Succès	Temps(S)	UB^*/z(E)
15	1-2	96.3%	23.7s	100.0%	**100.0%**	5.0s	X
	3-5	70.4%	52.5s	85.2%	**84.0%**	41.2s	99.4%
($m = 15$)		80.7%	38.8s	86.3%	**90.4%**	25.2s	99.4%
25	1-2	70.4%	39.3s	100.0%	**74.1%**	21.7s	99.9%
	3-5	44.4%	61.1s	94.1%	**45.7%**	44.8s	98.6%
($m = 25$)		54.8%	49.9s	95.6%	**57.0%**	32.8s	98.9%
(*Total*)		67.8%	43.3s	92.8%	**73.7%**	28.1s	99.0%

		(*Coupe* : $[MP] \hookrightarrow F.P.$)			(*Coupe* : $[MP] \hookrightarrow F.P.$) avec coupes initiales		
m	$\|C\|$	Succès	Temps(S)	UB^*/z(E)	Succès	Temps(S)	UB^*/z(E)
15	1-2	92.6%	15.3s	100.0%	**96.3%**	5.7s	100.0%
	3-5	76.5%	43.0s	99.1%	**88.9%**	39.8s	99.9%
($m = 15$)		83.0%	30.6s	99.2%	**91.9%**	25.5s	99.9%
25	1-2	**68.5%**	31.3s	99.9%	**68.5%**	18.1s	100.0%
	3-5	**50.6%**	57.9s	97.2%	**50.6%**	58.5s	99.5%
($m = 25$)		**57.8%**	45.3s	98.0%	**57.8%**	39.3s	99.7%
(*Total*)		70.4%	36.6s	98.4%	**74.8%**	30.8s	99.7%

TABLE 6.8 – Méthode de décomposition et génération de coupes

Nous constatons ensuite que la variante *exacte* (*Coupe*) présente des résultats similaires en termes de pourcentage de succès, mais légèrement meilleurs en termes de temps calcul aux deux variantes *approchées*. Ceci s'explique en partie par le fait que les deux techniques *approchées* doivent justifier de l'optimalité d'une solution en

vérifiant, à la dernière itération, que le problème maître (surcontraint par les coupes de borne et de réalisabilité ajoutées en cours de processus) ne possède plus aucune solution réalisable. Cette dernière itération s'avère assez coûteuse en pratique.

6.6.6 Synthèse comparative

Afin de mieux juger des performances relatives de chacune des méthodes exactes (proposées dans ce manuscrit) pour résoudre notre problématique couplant planification d'agents et ordonnancement de production, nous proposons ici deux tableaux comparatifs exprimant les performances de chaque technique sur des instances où chacune converge vers l'optimum.

Un premier tableau 6.9 s'intéresse aux 181 instances (67% de l'ensemble des instances) résolues optimalement par les trois méthodes exactes : résolution via le solveur MIP (MIP), décomposition de Benders ($Benders$) et la variante *exacte* de la génération de coupes ($Coupe$). Par souci de clarté, nous ne reportons que les résultats de la variante *exacte* de la génération de coupes car, comme on l'a vu précédemment, elle est plus efficace (dans un cadre de résolution exacte du problème) que les deux variantes *approchées*.

Un second tableau 6.10 permet de juger de la valeur ajoutée directe des travaux menés, sur cette première problématique, durant la thèse. On compare en effet ici les résultats obtenus via des techniques connues de la littérature (solveur MIP et décomposition de Benders) avec ceux obtenus par une méthode nouvelle : une technique de décomposition et de génération de coupes initialisée par des inégalités *énergétiques*. Le tableau 6.10 synthétise ainsi les résultats obtenus sur les 107 instances (38% de l'ensemble des instances) que résolvent optimalement ces trois méthodes exactes.

			(MIP) avec coupes initiales	(Benders) avec coupes initiales			(Coupe) avec coupes initiales		
m	$\|C\|$	#instances	Temps	Temps	#coupes de réalisabilité	# autres coupes initiales	Temps	#coupes de réalisabilité	# autres coupes initiales
15	1-2	52	9.5s	4.6s	4.5	2.8	**3.3s**	3.2	3.6
	3-5	63	**13.4s**	23.1s	38.5	28.8	24.4s	27.0	25.0
$(m = 15)$		115	**11.6s**	14.7s	23.2	17.0	14.8s	16.2	15.3
25	1-2	32	33.0s	**7.3s**	4.8	2.6	**7.3s**	3.9	2.0
	3-5	34	29.3s	**23.1s**	33.8	15.8	23.8s	18.3	13.7
$(m = 25)$		66	31.2s	**14.9s**	18.9	9.0	15.3s	10.9	7.7
$(Total)$		181	18.8s	**14.8s**	21.6	14.1	15.0s	14.3	12.5

TABLE 6.9 – Méthodes exactes

			(MIP)	(Benders)		(Coupe) avec coupes initiales		
m	$\|C\|$	#instances	Temps	Temps	#coupes de réalisabilité	Temps	#coupes de réalisabilité	# autres coupes initiales
15	1-2	41	23.7s	20.1s	24.5	**3.5s**	3.7	3.5
	3-5	22	**14.6s**	42.0s	77.0	14.8s	18.2	23.8
$(m = 15)$		63	20.6s	27.8s	42.8	**7.4s**	8.8	10.6
25	1-2	23	22.8s	22.4s	17.6	**2.9s**	2.9	1.3
	3-5	21	47.3s	58.3s	61.7	**19.6s**	8.1	8.4
$(m = 25)$		44	34.5s	39.5s	38.6	**10.9s**	5.4	4.7
$(Total)$		107	26.2s	32.6s	41.1	**8.9s**	7.4	8.2

TABLE 6.10 – Valeur ajoutée de la thèse pour la résolution exacte de la problématique

Un premier examen rapide du tableau 6.9 peut laisser croire à une équivalence, en termes de performance, entre la décomposition de Benders (*Benders*) et la génération de coupes (*Coupe*). Les temps de calcul sont en effet semblables (quelque soient les paramètres des instances) et légèrement meilleurs que ceux du solveur MIP. Comme de plus, ces deux méthodes présentent indépendamment des taux de résolution similaires (environ 74%), on pourrait précipitamment conclure à une certaine similarité. Il n'en est en réalité rien car si on s'intéresse de plus près à ces deux techniques (qui dominent toutes deux le solveur MIP), on s'aperçoit tout d'abord que sur l'ensemble des instances résolues par la décomposition de Benders et par la technique de coupes (en omettant ainsi la comparaison avec le solveur MIP), la technique de coupes est en moyenne plus rapide d'environ 12%. De plus, force est de constater que les bons résultats de la décomposition de Benders sont obtenus en grande partie en raison de la forte efficacité des coupes initiales. Rappelons en effet que sans les coupes initiales, la décomposition de Benders ne résout que 46.3% des instances alors que la technique de coupes en résout 56.3%. Les coupes initiales s'avèrent par conséquent ici d'une efficacité si grande que leur usage lisse les résultats purs des différentes techniques. Un examen plus approfondi des résultats amène dont à conclure que la technique de coupes est la méthode exacte de résolution la plus performante pour cette problématique.

Le tableau 6.10 démontre de manière assez éloquente que les diverses investigations menées lors de cette thèse pour la résolution propre de cette première problématique intégrant planification d'agents et ordonnancement de production ont permis de mettre en place une technique de résolution particulièrement efficace. La technique de coupes initialisée par les inégalités *énergétiques* est en effet de l'ordre de 3 fois plus rapide que les méthodes génériques (décrites dans la littérature) expérimentées ici.

Conclusion

Nous nous sommes intéressés dans cette partie du mémoire à une première problématique particulière couplant planification d'agents et ordonnancement de production.

Après avoir décrit le problème, nous avons proposé deux formalisations sous forme de Programmes Linéaires en Nombres Entiers. Nous nous sommes ensuite attachés à fournir deux bornes inférieures par décomposition lagrangienne. Les résultats obtenus se sont malheureusement avérés décevants. Nous avons après cela proposé deux techniques exactes de génération de coupes, une connue de la littérature (décomposition de Benders) et une autre plus singulière. Notre méthode de coupes, dont les résultats expérimentaux sont les plus probants, s'est révelée être particulièrement adaptée à la problématique. Elle présente de plus l'avantage d'être déclinable en une version approchée fournissant très rapidement des solutions réalisables de qualité en cours de processus ; on notera pour cette variante l'intérêt tout particulier de l'heuristique de Feasibility Pump pour la résolution heuristique du problème de *sac à dos multi-choix multidimensionnel*. Des inégalités déterminables en pré-process de toute méthode de résolution et utilisant le raisonnement *énergétique* ont enfin été développées. On a démontré, au travers des résultats numériques éloquents, qu'elles présentaient un intérêt significatif pour chacune des méthodes de résolution.

En perspective de ces travaux, de nombreux améliorations et prolongements intéressants demeurent à explorer.

Une première piste consisterait à résoudre incrémentalement les programmes maîtres de notre méthode de coupe. Une tentative reprenant cette idée a été menée durant cette thèse, nous avons en effet tenté de résoudre la problématique globale via un Branch-and-Bound (avec différents schémas de branchement) utilisant les fondements de notre technique de coupes et générant les inégalités de réalisabilité à la volée dans chaque branche étudiée. Les résultats préliminaires de cette approche se sont malheureusement montrés peu convaincants, nous avons donc préféré omettre sa présentation dans ce mémoire.

Une autre amélioration a priori prometteuse serait d'hybrider, à l'instar des travaux de Hooker et al. [Hoo05], nos techniques à des outils de programmation par contraintes.

Pour prolonger nos travaux, on peut aussi mentionner qu'il serait intéressant de

tester nos techniques de coupes sur d'autres problématiques intégrant planification d'agents et ordonnancement de production. Ce prolongement a lui aussi été considéré durant la thèse ; il fait d'ailleurs justement l'objet de la seconde partie de ce manuscrit.

Deuxième partie

Résolution d'un problème de Job-Shop intégrant des contraintes de Ressources Humaines

Introduction

Dans cette partie du mémoire, nous nous intéressons à une nouvelle problématique intégrant planification d'agents et ordonnancement de production.

La motivation initiale à cette seconde étude était de tester les méthodes de génération de coupes sur des problématiques plus complexes. Les techniques de décomposition et génération de coupes s'étaient en effet avérées si efficaces lors de nos premiers travaux qu'il semblait tout naturel de les expérimenter dans un cadre d'application plus général.

Cette seconde étude diffère de la première en raison de sa complexité intrinsèque. En effet, alors que le problème d'ordonnancement considéré précédemment se résolvait en un temps polynomial (problème de *flot maximal*), nous résolvons ici un problème d'ordonnancement *NP-complet*, à savoir la version décisionnelle d'un problème de job-shop.

Une autre motivation pour ces nouveaux travaux consistait également à pouvoir appliquer nos techniques de coupes à des instances connues de la littérature. La problématique intégrant planification d'agents et ordonnancement de production étudiée correspond ici à un cas particulier du problème considéré par Artigues et al. dans [AGRV09].

Après avoir présenté, dans ce chapitre introductif, le problème central d'ordonnancement de production qu'est le job-shop, nous définissons précisément le problème global intégrant planification d'agents et ordonnancement de production étudié durant la seconde partie de cette thèse.

Nous définissons après cela une formalisation de notre problème sous forme d'un Programme Linéaire en Nombres Entiers.

Nous poursuivons ensuite notre présentation par l'étude d'un problème particulier de job-shop : le problème de job-shop à périodes fixées d'inactivité. Ce problème étant à la base de nos techniques de résolution, il nous semble opportun de le présenter en préambule de ces dernières.

Un chapitre dédié à la présentation de techniques de génération d'inégalités valides pour toute méthode de résolution est proposé après cela.

Nous exposons ensuite les caractéristiques et fonctionnement d'une méthode exacte de décomposition et génération de coupes, analogue à la technique de coupes qui s'était avérée efficace lors de la première étude de cette thèse. Une seconde méthode exacte est présentée après ; elle est basée sur une méthode arborescente reprenant les concepts de la technique de coupes, à savoir la même décomposition du problème global en deux sous-problèmes et une génération identique d'inégalités de réalisabilité en cours de processus.

Nous discutons enfin de la performance de chaque technique en comparant leurs résultats numériques obtenus sur les instances compatibles d'Artigues et al. [AGRV09].

1 Présentation d'un problème de job-shop

1.1 Cadre général

Dans un problème de job-shop classique, n jobs doivent s'exécuter sur m machines selon une séquence opératoire propre à chacun.

L'exécution d'un job sur une machine est appelée opération. Les opérations ne peuvent pas être interrompues (préemption interdite) et les machines (indépendantes les unes des autres) exécutent au plus une opération à la fois.

Dans le cadre général, le critère d'optimisation attaché à un problème de job-shop est une fonction d'un certain nombre de paramètres pour ce problème. Pour ne citer que quelques exemples de fonctions-objectifs, on peut mentionner la minimisation de la somme pondérée des tâches en retard, la minimisation de la somme pondérée des tâches en avance, la minimisation de la durée totale de l'ordonnancement (aussi appelé *makespan*), etc.

Dans notre cas d'étude, nous ne considérerons que le critère *durée totale de l'ordonnancement*. Les séquencements des opérations sur les différentes machines sont alors inconnus et doivent être déterminés dans le but de minimiser la durée totale de l'ordonnancement.

Le problème de job-shop a été démontré *NP-complet* au sens fort par Rinnooy Kan dans [RK76] et Garey et Johnson dans [GJ79]. La plupart des méthodes efficaces proposées pour sa résolution sont du type procédures arborescentes. Nous nous référons à [CPPR04] pour une présentation exhaustive ainsi que pour un état de l'art sur le problème de job-shop.

1.2 Cadre de notre étude

Dans le cas précis de notre étude, la durée totale de l'ordonnancement (notons-la C_{max}) est connue. Cette démarche entre dans le cadre d'une politique industrielle où l'activité est planifiée selon des périodes fixées (une semaine par exemple) et les livraisons en produits finis sont toutes réalisées à échéance de la période considérée (à la fin de la semaine). Dans un tel mode de fonctionnement manufacturier, le décideur cherche alors à ordonnancer l'ensemble des opérations de telle manière que chaque job soit exécuté entièrement avant la fin de la période de planification choisie.

Ce problème spécifique consiste en la version décisionnelle d'un problème classique de job-shop ayant pour critère d'optimisation la durée totale de l'ordonnancement. On cherche ici à répondre à la question : *Existe-t-il un ordonnancement permettant la réalisation complète de chaque job avant C_{max} ?*

Le critère d'optimisation de la durée totale de l'ordonnancement devient alors une contrainte du problème.

2 Formulation de la problématique de notre étude

Nous nous plaçons dans le cadre d'un atelier de type job-shop à haute technicité qui fait intervenir des contraintes de qualification pour les opérateurs affectés au pilotage des machines. Plus précisément, nous considérons une unité manufacturière dans laquelle la production à réaliser requiert divers types de machines dans des séquences variées. Chaque machine nécessite pour son utilisation la présence d'un agent spécialement qualifié à son pilotage. Les ressources humaines sont également assujetties à des contraintes légales restreignant leur disponibilité. La production doit être entièrement ordonnancée et le critère d'optimisation retenu est la minimisation des coûts salariaux.

Ce problème coïncide avec le cas particulier des instances *correspondance activité-ressource* d'Artigues et al. [AGRV09]. Pour faciliter la comparaison, nous nous efforcerons d'utiliser les mêmes notations par la suite.

2.1 Horizon de planification

L'atelier fonctionne suivant une logique de travail cyclique (par exemple, travail dit *en 3 × 8*). L'horizon de planification s'étend ainsi sur un ensemble $\{s\}_{s=0,\dots,\sigma-1}$ de σ tranches horaires consécutives. La durée effective d'une tranche horaire est fixe et est notée π pour un horizon $H = \sigma \cdot \pi$.

2.2 Employés

Chaque instance fait intervenir un ensemble E de μ employés diversement qualifiés. Pour chaque employé $e \in E$, on dispose de l'ensemble A_e des machines sur lesquelles il peut intervenir ainsi que de l'ensemble $\mathcal{T}_e$ des tranches horaires pour lesquelles il est disponible. Un employé ne peut pas changer de machine en cours de poste : un employé affecté à une machine k durant une tranche horaire s travaille donc sur k durant la totalité de s. Des contraintes légales assujettissent de plus chaque employé à ne pouvoir travailler qu'au maximum une fois parmi trois tranches horaires consécutives, et ce afin de coller au plus près à la logique de travail cyclique.

Un coût d'affectation c_{eks} est attribué à chaque employé e travaillant sur la machine k durant la tranche horaire s. Le critère d'optimisation de notre problématique est alors basé sur ces coûts d'affectation des employés aux machines et aux tranches horaires.

Une solution à la problématique de planification d'agents du problème global est donnée par une séquence de couples (machine utilisée - tranche horaire travaillée) propre à chaque employé ; la somme sur chaque employé des coûts de leur séquence définissant le coût total de la solution.

2.3 Job-Shop

Le problème d'ordonnancement de production étudié ici est la variante décisionnelle d'un problème classique de job-shop dont le critère d'optimisation est la durée totale de l'ordonnancement.

On considère un ensemble J de n jobs et un ensemble M de m machines. Chaque job i est caractérisé par une séquence propre d'opérations $\{O_{ij}\}_{j=1,\dots,m}$ non préemptives. L'opération O_{ij} s'effectue sur une machine m_{ij} donnée et sa durée d'exécution est notée p_{ij}. Pour plus de commodité, nous utilisons la notation classique ρ_{ik} pour désigner la durée de l'opération du job i qui s'exécute sur la machine k.

L'objectif du problème de job-shop est alors de déterminer un ordonnancement réalisable de l'ensemble des n jobs, tel que la durée totale d'ordonnancement soit inférieure ou égale à une date $C_{max} \leq H$ connue.

2.4 Objectif

L'objectif du problème global auquel nous nous intéressons est de déterminer un ordonnancement réalisable pour le problème de job-shop en affectant une séquence de couples (machine utilisée - tranche horaire travaillée) à chaque employé, de telle sorte que chaque job soit exécuté avant C_{max} et que les besoins en main d'œuvre (effectif et compétence) soient remplis au moindre coût.

2.5 Structure du problème

Comme dans la première partie de ce mémoire, il y a clairement ici deux niveaux de décision. Il faut tout d'abord établir un planning de travail pour chaque employé, c'est-à-dire décider pour chaque employé sur quelles machines et durant quelles tranches horaires il travaillera. Il faut ensuite déterminer un ordonnancement complet compatible avec les effectifs prédéfinis.

3 Problématique étudiée par Artigues et al.

Comme nous l'avons mentionné en introduction de ce chapitre, une des motivations à l'étude de cette problématique était de pouvoir expérimenter nos méthodes de résolution sur des instances de la littérature. Nous nous sommes donc concentrés sur le problème défini ci-dessus car il correspondait à un des problèmes étudiés par Artigues et al. [AGRV09].

Nous détaillons ci-après la nature exacte du problème décrit dans [AGRV09]. Nous présentons ensuite les techniques de résolution définies par les auteurs de cet article.

3.1 Nature du problème

Artigues et al. s'intéressent à l'étude d'un problème couplant planification d'agents et ordonnancement de production.

D'un côté, nous devons résoudre un cas particulier de problème de gestion d'emplois du temps consistant à affecter au moindre coût des séquences de tranches horaires de présence ou d'absence à des agents différemment qualifiés ; chaque agent étant soumis à des contraintes législatives l'obligeant à ne pouvoir travailler qu'au maximum une fois parmi trois tranches horaires consécutives.

D'un autre côté, un problème de production de type job-shop, dont la fonction-objectif porte sur la minimisation de la durée totale de l'ordonnancement, doit être résolu ; chaque job étant composé de tâches requérant chacune, pour son exécution, un ou plusieurs employés diversement qualifiés.

La problématique de [AGRV09] est par conséquent très similaire à celle étudiée dans la seconde partie du manuscrit de cette thèse. Des instances dites de *correspondance activité-ressource* correspondant exactement au problème auquel nous nous intéressons sont même proposées dans cet article. Nous nous appuyons sur ces dernières pour expérimenter les techniques de résolution que nous présentons par la suite.

3.2 Techniques de résolution

Dans leurs travaux, Artigues et al. proposent deux méthodes exactes hybridant Programmation Linéaire en Nombres Entiers et Programmation Par Contraintes. Chacune de ces deux techniques est basée sur une formalisation PPC, associée à la relaxation continue de contraintes linéaires manipulant des variables entières. La relaxation continue de chaque contrainte linéaire est concrètement intégrée au modèle PPC au travers d'une contrainte globale. Des techniques de filtrage par coûts réduits sont de plus utilisées.

Chapitre 7

Formalisation indexée sur le temps

Nous avons défini dans le chapitre précédent les caractéristiques de la seconde problématique étudiée lors de cette thèse. Nous en présentons ici une formalisation en termes de programmes linéaires. Nous débutons ce nouveau chapitre en introduisant les données (et leur notation [6]) en entrées du problème, et proposons ensuite un modèle sous la forme d'un Programme Linéaire en Nombres Entiers.

7.1 Données

Les données suivantes constituent les entrées de notre problème ; on peut les différencier selon deux catégories, à savoir les données inhérentes à la planification des agents d'une part et les données concernant l'ordonnancement de production d'autre part.

- ***Planification d'agents***
  - E est l'ensemble des μ employés
  - S est l'ensemble des σ tranches horaires utiles à la planification des employés
  - π est la durée de chaque tranche horaire
  - $H = \sigma \cdot \pi$ désigne alors l'horizon de planification
  - A_e correspond à la liste des machines sur lesquelles peut intervenir l'employé $e \in E$
  - T_e correspond à la liste des tranches horaires pour lesquelles l'employé $e \in E$ est disponible
  - c_{eks} représente le coût d'affectation de l'employé e à la machine k durant la tranche horaire s
- ***Ordonnancement de production***
  - J est l'ensemble des n jobs à ordonnancer
  - M représente l'ensemble des m machines
  - C_{max} est la borne supérieure autorisée pour la durée totale de l'ordonnancement

6. Pour faciliter la comparaison des travaux, nous utilisons les notations utilisées par Artigues et al. dans [AGRV09]

- ρ_{ik} est la durée de l'opération du job i s'exécutant sur la machine k
- m_{ij} est la machine utilisée lors de la $j^{\text{ème}}$ opération de la séquence opératoire du job i
- r_{ik} et d_{ik} sont respectivement la date de démarrage au plus tôt et la date échue de l'opération du job i s'exécutant sur la machine k. r_{ik} et d_{ik} sont définies en pré-process par les équations de récurrence (7.1) et (7.2).

$$\begin{cases} r_{ik} = 0 & i = 1, \ldots, n \quad k = m_{i1} \\ r_{il} = r_{ik} + \rho_{ik} & i = 1, \ldots, n \quad j = 1, \ldots, m-1 \quad k = m_{ij} \quad l = m_{i(j+1)} \end{cases} \tag{7.1}$$

L'expression (7.1) contraint chaque opération d'un job i donné à ne pouvoir débuter avant que l'opération la précédant dans la séquence opératoire de i ne puisse être terminée.

$$\begin{cases} d_{ik} = C_{max} & i = 1, \ldots, n \quad k = m_{im} \\ d_{ik} = d_{il} - \rho_{il} & i = 1, \ldots, n \quad j = 1, \ldots, m-1 \quad k = m_{ij} \quad l = m_{i(j+1)} \end{cases} \tag{7.2}$$

Les relations (7.2) indiquent que chaque opération d'un job i donné doit être terminée avant que l'opération lui succédant dans la séquence opératoire de i ne doive être nécessairement commencée.

7.2 Variables de décision

Afin de modéliser notre problème sous la forme d'un Programme Linéaire en Nombres Entiers, deux variables binaires de décision sont introduites :

- $x_{eks} = 1$ si l'employé e est affecté à la machine k durant la tranche horaire s, 0 sinon
- $y_{ikt} = 1$ si le job i commence son exécution sur la machine k à l'instant t, 0 sinon

7.3 Programme Linéaire

Forts des données et des variables de décision présentées ci-dessus, nous sommes désormais en mesure de proposer une formalisation de notre problématique sous la forme d'un programme linéaire en variables binaires.

Par souci de clarté, nous séquençons la présentation de notre modèle. Nous introduisons dans un premier temps la fonction-objectif du problème. Nous détaillons ensuite les contraintes spécifiques inhérentes au problème de planification d'agents considéré. Les contraintes propres au problème d'ordonnancement de production (job-shop) sont ensuite exposées. Nous nous intéressons après cela aux contraintes couplantes de la problématique intégrée. Enfin, un programme linéaire complet récapitulatif est proposé.

7.3.1 Fonction-objectif

Le critère d'optimisation de la problématique étudiée est basé sur la minimisation des coûts salariaux. La relation (7.3) exprime cet objectif, les coûts salariaux étant induits par les affectations d'employés aux machines et aux tranches horaires.

$$\min \Theta = \sum_{e \in E} \sum_{k \in A_e} \sum_{s \in T_e} c_{eks} \cdot x_{eks} \tag{7.3}$$

7.3.2 Contraintes spécifiques au problème de planification d'agents

Les contraintes propres au problème de planification d'agents considéré dans notre étude sont exprimées ci-après.

$$\sum_{k \notin A_e} \sum_{s=0}^{\sigma} x_{eks} = 0 \qquad e = 1, \ldots, \mu \tag{7.4}$$

$$\sum_{k \in A_e} \sum_{s \notin \mathcal{T}_e} x_{eks} = 0 \qquad e = 1, \ldots, \mu \tag{7.5}$$

$$\sum_{k \in A_e} (x_{eks} + x_{ek(s+1)} + x_{ek(s+2)}) \leq 1 \quad e = 1, \ldots, \mu \quad s = 0, \ldots, \sigma - 3 \tag{7.6}$$

$$x_{eks} \in \{0, 1\} \qquad e = 1, \ldots, \mu \quad k = 1, \ldots, m \quad s = 0, \ldots, \sigma - 1 \tag{7.7}$$

Les contraintes (7.4) précisent qu'un employé ne peut être affecté que sur une machine sur laquelle il est qualifié.

Chaque employé $e \in E$ est de plus contraint, par (7.5), à ne travailler que sur des tranches horaires pour lesquelles il est disponible.

Enfin, la contraintes légale, exprimée par (7.6), oblige chaque employé à ne travailler au maximum qu'une tranche horaire parmi trois tranches consécutives.

7.3.3 Contraintes spécifiques au problème de Job-Shop

Le problème d'ordonnancement de production de la problématique globale est lui aussi caractérisé par des contraintes qui lui sont spécifiques. Ces dernières sont indiquées ci-après.

$$\sum_{t=0}^{d_{ik}-\rho_{ik}} t \cdot y_{ikt} + \rho_{ik} \leq C_{max} \quad i = 1, \ldots, n \quad k = m_{im} \tag{7.8}$$

$$\sum_{t=r_{ik}}^{d_{ik}-\rho_{ik}} y_{ikt} = 1 \quad i = 1, \ldots, n \quad k = 1, \ldots, m \tag{7.9}$$

$$\sum_{t=0}^{r_{ik}} y_{ikt} + \sum_{t=d_{ik}-\rho_{ik}+1}^{C_{max}} y_{ikt} = 0 \quad i = 1, \ldots, n \quad k = 1, \ldots, m \tag{7.10}$$

$$\sum_{u=r_{ik}+\rho_{ik}}^{t} y_{ilu} - \sum_{u=r_{ik}}^{t-\rho_{ik}} y_{iku} \leq 0 \quad i = 1, \ldots, n \quad j = 1, \ldots, m-1$$
$$k = m_{ij} \quad l = m_{i(j+1)}$$
$$t = r_{ik} + \rho_{ik}, \ldots, d_{il} - \rho_{il} \tag{7.11}$$

$$\sum_{i=1}^{n} \sum_{u=\max(r_{ik}, t-\rho_{ik}+1)}^{\min(d_{ik}-\rho_{ik}, t)} y_{iku} \leq 1 \quad k = 1, \ldots, m \quad t = 0, \ldots, C_{max} \tag{7.12}$$

$$y_{ikt} \in \{0, 1\} \quad i = 1, \ldots, n \quad k = 1, \ldots, m$$
$$t = 0, \ldots, C_{max} \tag{7.13}$$

Les inégalités (7.8) assurent que chaque job est terminé avant la date de fin d'ordonnancement C_{max} désirée par le décideur.

Les contraintes (7.9) et (7.10) spécifient pour leur part que toute opération doit être exécutée dans sa fenêtre d'exécution.

Le respect des séquences opératoires de chaque job est contrôlé par (7.11).

Enfin, (7.12) exprime les contraintes de disponibilité des machines, c'est-à-dire : au plus une opération en cours sur chaque machine à un instant donné.

7.3.4 Contraintes couplantes

Notre problématique générale intégrant planification d'agents et ordonnancement de production, il est logique que des contraintes couplant ces deux niveaux de décision interviennent dans notre modèle. Nous présentons ces contraintes couplantes dans cette partie de l'exposé.

Les contraintes couplantes (7.14) assurent la présence d'un employé sur chaque machine et à chaque instant où une opération est en cours d'exécution.

$$\sum_{e \in E} x_{eks} - \sum_{i=1}^{n} \sum_{u=\max(r_{ik}, t-\rho_{ik}+1)}^{\min(d_{ik}-\rho_{ik}, t)} y_{iku} \geq 0 \quad k = 1, \ldots, m \quad t = 0, \ldots, C_{max}$$
$$s = \lfloor t/\pi \rfloor \tag{7.14}$$

7.3.5 Programme Linéaire

Nous récapitulons les éléments mentionnés dans ce chapitre en présentant ci-dessous une modélisation complète $[PL]$ de notre problématique sous la forme d'un programme linéaire en variables binaires.

$$
\begin{aligned}
[PL]: \qquad \min \Theta &= \sum_{e\in E}\sum_{k\in A_e}\sum_{s\in T_e} c_{eks}\cdot x_{eks} \\
\sum_{k\notin A_e}\sum_{s=0}^{\sigma} x_{eks} &= 0 && e=1,\dots,\mu \\
\sum_{k\in A_e}\sum_{s\notin \mathcal{T}_e} x_{eks} &= 0 && e=1,\dots,\mu \\
\sum_{k\in A_e}\left(x_{eks}+x_{ek(s+1)}+x_{ek(s+2)}\right) &\le 1 && e=1,\dots,\mu \\
& && s=0,\dots,\sigma-3 \\
\sum_{t=0}^{d_{ik}-\rho_{ik}} t\cdot y_{ikt}+\rho_{ik} &\le C_{max} && i=1,\dots,n \quad k=m_{im} \\
\sum_{t=r_{ik}}^{d_{ik}-\rho_{ik}} y_{ikt} &= 1 && i=1,\dots,n \quad k=1,\dots,m \\
\sum_{t=0}^{r_{ik}} y_{ikt}+\sum_{t=d_{ik}-\rho_{ik}+1}^{C_{max}} y_{ikt} &= 0 && i=1,\dots,n \quad k=1,\dots,m \\
\sum_{u=r_{ik}+\rho_{ik}}^{t} y_{ilu}-\sum_{u=r_{ik}}^{t-\rho_{ik}} y_{iku} &\le 0 && i=1,\dots,n \quad j=1,\dots,m-1 \\
& && k=m_{ij} \quad l=m_{i(j+1)} \\
& && t=\rho_{ik}+p_{ik},\dots,d_{il}-\rho_{il} \\
\sum_{i=1}^{n}\sum_{u=\max(r_{ik},t-\rho_{ik}+1)}^{\min(d_{ik}-\rho_{ik},t)} y_{iku} &\le 1 && k=1,\dots,m \quad t=0,\dots,C_{max} \\
\sum_{e\in E} x_{eks}-\sum_{i=1}^{n}\sum_{u=\max(r_{ik},t-\rho_{ik}+1)}^{\min(d_{ik}-\rho_{ik},t)} y_{iku} &\ge 0 && k=1,\dots,m \quad t=0,\dots,C_{max} \\
& && s=\lfloor t/\pi \rfloor \\
x_{eks}\in\{0,1\} & && e=1,\dots,\mu \quad k=1,\dots,m \\
& && s=0,\dots,\sigma-1 \\
y_{ikt}\in\{0,1\} & && i=1,\dots,n \quad k=1,\dots,m \\
& && t=0,\dots,C_{max}
\end{aligned}
$$

Chapitre 8

Job-shop à périodes fixées d'inactivité des machines

Nous nous intéressons, dans ce chapitre, à la présentation d'un problème de job-shop à périodes fixées d'inactivité des machines. Comme nous le verrons par la suite, ce problème est central pour nos techniques de résolution.

Nous en proposons tout d'abord une définition et détaillons ensuite les enjeux de l'étude ce type de problèmes pour notre problématique générale. Nous présentons après cela deux modélisations sous forme de programme linéaire, la première découlant naturellement de la définition du problème, et la seconde exploitant la transformation des périodes d'inactivité en jobs fictifs. Les techniques de résolution utilisées dans nos expérimentations sont enfin exposées.

8.1 Définition et enjeux

8.1.1 Définition

Définition 2 *Un problème de job-shop à périodes fixées d'inactivité des machines est défini comme un problème de job-shop pour lequel des périodes de non-utilisation, propres à chaque machine, doivent être planifiées. Aucune opération ne peut être exécutée durant les périodes d'inactivité d'une machine. Ces périodes sont connues à l'avance par le décideur, leur répartition sur chaque machine est donc une donnée du problème.*

8.1.2 Enjeux pour la problématique générale

Supposons, pour notre problématique générale, une solution de séquences de couples (machine utilisée - tranche horaire travaillée) affectée à chaque employé. En s'inspirant des notations de la formalisation de la problématique proposée en 7.3.5, cela revient à considérer une distribution de x fixée (notons-la $\bar{x}$).

Il est alors clair qu'il existe une solution réalisable du problème global, compatible avec le vecteur $\bar{x}$, si et seulement si on peut définir un ordonnancement de l'ensemble des jobs de telle manière qu'aucun job ne soit exécuté sur un couple (machine - tranche horaire) où aucun employé ne travaille selon $\bar{x}$.

Considérons alors un ensemble de couples (machine $\bar{k}$ - tranche horaire $\bar{s}$) pour lesquels $\bar{x}$ impose l'absence d'employés. De telles absences induiront, pour le problème de job-shop, des incapacités ponctuelles de production. Pour chacun de ces couples $(\bar{k}, \bar{s})$, la machine $\bar{k}$ sera inactive durant la période $\bar{s}$. Le problème d'ordonnancement contraint par $\bar{x}$ est alors un problème de job-shop à périodes fixées d'inactivité des machines.

8.2 Formalisation directe

Nous présentons ci-après une formalisation directe, en termes de programmes linéaires, d'un problème de job-shop soumis à des périodes fixées d'inactivité.

Par souci de clarté, nous utilisons les mêmes notations que celles énoncées dans le chapitre 7 pour la formalisation de la partie job-shop de la problématique générale.

8.2.1 Données

Les données constituant les entrées du problème peuvent être listées comme suit :

- ***Données communes à un problème de job-shop classique***
  - J est l'ensemble des n jobs à ordonnancer
  - m représente le nombre de machines
  - C_{max} est une borne supérieure autorisée pour la durée totale de l'ordonnancement
  - ρ_{ik} est la durée de l'opération du job i s'exécutant sur la machine k
  - m_{ij} est la machine utilisée lors de la $j^{\text{ème}}$ opération de la séquence opératoire du job i
  - r_{ik} et d_{ik} sont respectivement la date de démarrage au plus tôt et la date échue de l'opération du job i s'exécutant sur la machine k. r_{ik} et d_{ik} sont définies en pré-process par les équations de récurrence (7.1) et (7.2), présentées en 7.1.

- ***Données spécifiques à un problème de job-shop à périodes fixées d'inactivité***
  - $\Upsilon = \bigcup_{k=1}^{m} \Upsilon^k$ est la liste complète des périodes fixées d'inactivité, Υ^k représentant la liste des périodes d'inactivité de la machine k. Chaque période d'inactivité $\upsilon \in \Upsilon$ est définie telle que $\upsilon = [\![\alpha_\upsilon, \beta_\upsilon]\!]$, avec α_υ et β_υ les instants respectifs de début et de fin d'inactivité imposée.

8.2.2 Variables de décision

Nous définissons un groupe de variables binaires de décision y régulant les instants de démarrage de chaque job sur chaque machine :

– $y_{ikt} = 1$ si le job i commence son exécution sur la machine k à l'instant t, 0 sinon

8.2.3 Formalisation du problème d'existence

Une première modélisation [$JobShop_F$] en termes de programmes linéaires du problème d'existence d'un problème de job-shop soumis à un ensemble Υ de périodes fixées d'inactivité des machines peut alors être proposée :

[$JobShop_F$] : *Existe-t-il un ordonnancement vérifiant :*

$$\sum_{t=0}^{d_{ik}-\rho_{ik}} t \cdot y_{ikt} + \rho_{ik} \leq C_{max} \quad i = 1, \ldots, n \quad k = m_{im} \tag{8.1}$$

$$\sum_{t=r_{ik}}^{d_{ik}-\rho_{ik}} y_{ikt} = 1 \quad i = 1, \ldots, n \quad k = 1, \ldots, m \tag{8.2}$$

$$\sum_{t=0}^{r_{ik}} y_{ikt} + \sum_{t=d_{ik}-\rho_{ik}+1}^{C_{max}} y_{ikt} = 0 \quad i = 1, \ldots, n \quad k = 1, \ldots, m \tag{8.3}$$

$$\sum_{u=r_{ik}+\rho_{ik}}^{t} y_{ilu} - \sum_{u=r_{ik}}^{t-\rho_{ik}} y_{iku} \leq 0 \quad i = 1, \ldots, n \quad j = 1, \ldots, m-1$$
$$k = m_{ij} \quad l = m_{i(j+1)}$$
$$t = r_{ik} + \rho_{ik}, \ldots, d_{il} - \rho_{il} \tag{8.4}$$

$$\sum_{i=1}^{n} \sum_{u=\max(r_{ik}, t-\rho_{ik}+1)}^{\min(d_{ik}-\rho_{ik}, t)} y_{iku} \leq 1 \quad k = 1, \ldots, m \quad t = 0, \ldots, C_{max} \tag{8.5}$$

$$\sum_{i=1}^{n} \sum_{u=\max(r_{ik}, \alpha_\upsilon-\rho_{ik})}^{\beta_\upsilon} y_{ikt} = 0 \quad k = 1, \ldots, m \quad \upsilon \in \Upsilon^k \tag{8.6}$$

$$y_{ikt} \in \{0, 1\} \quad i = 1, \ldots, n \quad k = 1, \ldots, m$$
$$t = 0, \ldots, C_{max} \tag{8.7}$$

Les jeux de contraintes (8.1) à (8.5) sont communs à un problème classique de job-shop (cf. chapitre 7).

Les contraintes (8.6) sont les contraintes propres au problème de job-shop à périodes fixées d'inactivité : elles permettent d'interdire le travail sur les périodes d'inactivité de chaque machine.

8.3 Formalisation sous forme d'un problème classique de job-shop

Nous présentons dans cette section une modélisation alternative d'un problème de job-shop à périodes fixées d'inactivité. Nous exploitons ici la réécriture de chaque période d'inactivité d'une machine en un job fictif. Un problème classique de job-shop, composé de jobs réels (données du problème) et de jobs fictifs, est ainsi obtenu.

Le problème de job-shop étant un problème central en Théorie de l'Ordonnancement, son étude a donné lieu à de nombreuses publications dans la littérature. Des techniques de résolution efficaces sont donc connues pour cette classe de problème. Nous renvoyons le lecteur à [CPPR04] pour un état de l'art sur le job-shop. Notre motivation quant à déterminer cette seconde modélisation était donc conditionnée originellement par le fait de pouvoir appliquer ces méthodes, performantes pour les problèmes classiques de job-shop, à des problèmes de job-shop à périodes fixées d'inactivité des machines.

8.3.1 Modélisation d'une période d'inactivité par un job fictif

Proposition 8 *Interdire la production durant une période $\upsilon = [\![\alpha_\upsilon, \beta_\upsilon]\!]$ sur la machine k est équivalent à ordonnancer un job fictif i_F composé de m opérations toutes de domaine d'exécution $[\![0, C_{max}]\!]$ et de durée nulle sauf une, l'opération s'exécutant sur k ; cette opération a pour domaine d'exécution $[\![\alpha_\upsilon, \beta_\upsilon]\!]$, est de durée $\rho_{i_F k} = (\beta_\upsilon - \alpha_\upsilon)$, et peut être arbitrairement placée à n'importe quelle position dans la séquence propre de i_F.*

Preuve.
Les opérations de domaine d'exécution $[\![0, C_{max}]\!]$ et de durée nulle composant i_F sont toujours exécutables. L'ordonnancement de i_F est donc exclusivement conditionné par l'ordonnancement de la seule opération de durée non nulle, à savoir l'opération s'exécutant sur k durant υ ; cette opération pouvant alors être arbitrairement placée à n'importe quelle position dans la séquence propre de i_F, par exemple à la première. En utilisant les notations d'un problème classique de job-shop mentionnées en 2.3, cela revient à noter cette opération $O_{i_F 1}$.

$O_{i_F 1}$ est de durée $(\beta_\upsilon - \alpha_\upsilon)$ et a pour domaine d'exécution $[\![\alpha_\upsilon, \beta_\upsilon]\!]$. Ordonnancer $O_{i_F 1}$ implique alors que la machine k soit réservée entièrement à l'exécution de $O_{i_F 1}$ durant υ. Aucun job réel du problème de job-shop ne peut donc être exécuté sur k durant υ. La machine k est donc concrètement inactive durant υ. ■

En créant un job fictif pour chaque période fixée d'inactivité $\upsilon \in \Upsilon$, on obtient un ensemble J_F de n_F jobs fictifs, avec $n_F = \text{card}\,\{\Upsilon\}$. L'objectif du problème d'ordonnancement consiste alors à ordonnancer l'ensemble des jobs réels J et l'ensemble J_F

des jobs fictifs. Le problème de job-shop à périodes fixées d'inactivité des machines est ainsi réécrit en un problème classique de job-shop.

8.3.2 Formalisation du problème d'existence

En utilisant les mêmes données, variables de décision et contraintes que précédemment, on peut alors proposer une formalisation alternative à $[JobShop_F]$, cette nouvelle formalisation étant celle d'un problème classique de job-shop :

$[JobShop'_F]$: *Existe-t-il un ordonnancement vérifiant :*

$$\sum_{t=0}^{d_{ik}-\rho_{ik}} t \cdot y_{ikt} + \rho_{ik} \leq C_{max} \qquad i = 1, \ldots, n+n_F \quad k = m_{im}$$

$$\sum_{t=r_{ik}}^{d_{ik}-\rho_{ik}} y_{ikt} = 1 \qquad i = 1, \ldots, n+n_F \quad k = 1, \ldots, m$$

$$\sum_{t=0}^{r_{ik}} y_{ikt} + \sum_{t=d_{ik}-\rho_{ik}+1}^{C_{max}} y_{ikt} = 0 \quad i = 1, \ldots, n+n_F \quad k = 1, \ldots, m$$

$$\sum_{u=r_{ik}+\rho_{ik}}^{t} y_{ilu} - \sum_{u=r_{ik}}^{t-\rho_{ik}} y_{iku} \leq 0 \quad i = 1, \ldots, n+n_F \quad j = 1, \ldots, m-1$$

$$k = m_{ij} \quad l = m_{i(j+1)}$$

$$t = r_{ik} + \rho_{ik}, \ldots, d_{il} - \rho_{il}$$

$$\sum_{i=1}^{n+n_F} \sum_{u=\max(r_{ik}, t-\rho_{ik}+1)}^{\min(d_{ik}-\rho_{ik}, t)} y_{iku} \leq 1 \quad k = 1, \ldots, m \quad t = 0, \ldots, C_{max}$$

$$y_{ikt} \in \{0, 1\} \qquad i = 1, \ldots, n+n_F \quad k = 1, \ldots, m$$

$$t = 0, \ldots, C_{max}$$

8.4 Techniques de résolution

La littérature sur le problème de job-shop est particulièrement abondante. Il est par conséquent difficile d'établir un état de l'art exhaustif de ce problème. L'objet de cette thèse ne portant pas directement sur la résolution du jobshop, nous renvoyons le lecteur aux travaux de Carlier et al. [CPPR04] afin d'apréhender le problème de job-shop de manière plus significative.

Nous utilisons dans notre étude le solveur dédié de job-shop proposé dans [Riv99]. Cet outil, basé sur une approche arborescente, s'appuie en partie sur les opérations d'arbitrage applicables aux problèmes d'ordonnancement disjonctifs [7] définies dans [CP94, CP95, P96, PPR97, BPR03, CPPR04, PR05].

7. Problèmes dans lequels certaines tâches se partagent une ou plusieurs ressources qui ne peuvent être consacrées qu'à une seule opération à un instant donné ; le problème de job-shop en est un exemple.

Dans nos expérimentations, les règles d'élimination sélectionnées sont :
- Arbitrage sur disjonction
- Arbitrage sur ensembles ascendants avec shaving

Le solveur de job-shop auquel nous avons recours renvoie trois types de réponse :
- Le problème de job-shop est réalisable
- Le problème de job-shop est irréalisable
- La réalisabilité -ou l'irréalisabilité- du problème de job-shop n'a pas pu être établie dans le temps imparti (1 seconde dans nos expérimentations)

Expérimentalement, dans les cas où la preuve de réalisabilité est nécessaire, le critère d'arrêt sur le temps limite de calcul est désactivé et le problème de job-shop résolu jusqu'à ce que la preuve (ou sa réfutation) soit faite.

Chapitre 9

Inégalités initiales valides

Nous décrivons, dans ce chapitre, trois groupes d'inégalités valides pour cette seconde problématique générale couplant planification d'agents et ordonnancement de production. À l'instar de la première étude menée au cours de cette thèse, ces inégalités sont générées en pré-process, indépendamment de la technique de résolution utilisée par la suite.

Nous exposons tout d'abord une première stratégie, dite de *probing*, qui vise à déterminer des sous-ensembles de couples (machine, tranche horaire) pour lesquels il existe des obligations de travail. Une seconde technique de génération d'inégalités, exploitant cette fois-ci le nombre minimal de tranches horaires travaillées sur chaque machine, est ensuite détaillée. Un troisième ensemble de coupes, complémentaire aux deux premiers, est enfin présenté.

9.1 Probing

Dans cette première stratégie de génération d'inégalités valides, nous cherchons à caractériser des sous-ensembles de couples (machine, tranche horaire) pour lesquels il existe des obligations de travail.

9.1.1 Description

La technique dite de *probing* utilisée consiste tout d'abord à interdire le travail sur un certain nombre de couples (machine, tranche horaire). Un ordonnancement réalisable de l'ensemble des jobs $i \in J$ vérifiant ces interdictions de travail est ensuite recherché. S'il n'en existe pas, cela signifie que l'ensemble des interdictions de travail proposé ne peut être respecté ; une coupe initiale de réalisabilité est alors générée.

Nous qualifions cette technique de *méthode de probing*, par analogie au terme *probing* utilisé pour la résolution de Programmes Linéaires en Nombres Entiers. Comme il est décrit dans [Sav94], les techniques de pré-processing dites de *probing*

s'appliquent à tout PLNE. Elles ont pour objet de rechercher les conséquences logiques déterminées après qu'on a fixé successivement des variables de décision à différentes valeurs de leur domaine de définition (par exemple, à 0 et/ou 1 pour des variables binaires). Par l'analyse des impacts de chacun de ces choix sur le modèle, il est parfois possible de déterminer la valeur optimale d'un certain nombre de variables de décision et ce, avant même de débuter la résolution directe du programme. Dans notre approche, on fixe au repos certains couples (machine, tranche horaire) puis on analyse la réalisabilité pour le problème d'ordonnancement de ces interdictions de travail. S'il s'avère qu'aucun ordonnancement compatible n'existe, on retire une information (une coupe de réalisabilité) que toute solution réalisable, et donc optimale, vérifie nécessairement.

9.1.2 Nature et résolution du problème d'ordonnancement surcontraint

Le problème d'ordonnancement surcontraint par les interdictions de travail sur certains couples (machine, tranche horaire) est, par définition, un problème de job-shop à périodes fixées d'inactivité des machines. On a vu, dans le chapitre 8, que ce problème pouvait être réécrit en un problème classique de job-shop en introduisant un ensemble de jobs fictifs, chaque job fictif correspondant à un couple (machine k, tranche horaire s) imposé au repos. Nous exploitons, dans nos expérimentations, cette réécriture et résolvons le problème résultant en appliquant les techniques de résolution d'un problème classique de job-shop décrites en 8.4.

9.1.3 Coupe de *probing*

Supposons un ensemble $\Upsilon = \cup_{k=1}^{m} \Upsilon^k$, Υ^k indiquant une liste de tranches horaires imposées arbitrairement au repos pour la machine k.

Si le problème induit de job-shop soumis à l'ensemble Υ de périodes fixées d'inactivité n'est pas réalisable, cela signifie que l'une au moins des tranches horaires de Υ imposées au repos doit être travaillée.

La coupe de *probing* (9.1) est alors une coupe valide pour le problème global.

$$\sum_{e \in E} \sum_{k \in A_e} \sum_{s \in T_e} \alpha_k^s \cdot x_{eks} \geq 1 \tag{9.1}$$

où $\alpha_k^s = 1$ si $s \in \Upsilon^k$, 0 sinon.

9.1.4 Expérimentations

Dans nos expérimentations, nous testons individuellement l'interdiction de travail sur chaque couple $(k, s) \in \{M \times S\}$. L'ensemble Υ des tranches horaires imposées au repos est alors un singleton composé d'un unique couple (machine, tranche horaire). Les coupes de *probing* ainsi générées forcent le travail sur un unique couple $(k, s) \in \{M \times S\}$. On dira par la suite que le couple (k, s) est *fixé par probing*, conformément à la définition 3.

Définition 3 *Tout couple $(k, s) \in \{M \times S\}$ pour lequel la technique de probing prouve que le travail est obligatoire est dit fixé par probing ; la machine k est en activité durant la tranche horaire s dans toute solution réalisable.*

On cherche ensuite à générer une coupe de *probing* pour chaque ensemble composé de deux couples (k, s) et (k, s') -avec $k \in M$ et $(s, s') \in S^2$; $s \neq s'$- tels que (k, s) et (k, s') sont tous deux non fixés par probing.

9.2 Nombre minimal de tranches horaires travaillées sur chaque machine

Nous présentons dans cette section une seconde technique de génération de coupes initiales de réalisabilité. On s'intéresse ici à exprimer une borne inférieure du nombre de tranches horaires devant être nécessairement travaillées sur chaque machine.

9.2.1 Description

Nous calculons, pour chaque machine $k \in M$, une borne inférieure du nombre minimal de tranches horaires nécessairement travaillées de trois manières distinctes ; trois valeurs LB_k^1, LB_k^2 et LB_k^3 sont ainsi obtenues. Une coupe de réalisabilité obligeant le travail, sur la machine k, durant au moins LB_k (où $LB_k = \max(LB_k^1, LB_k^2, LB_k^3)$) tranches horaires est alors générée.

9.2.2 Borne inférieure LB_k

Considérons une machine $k \in M$. Nous définissons dans cette section trois bornes inférieures (LB_k^1, LB_k^2 et LB_k^3) du nombre minimal de tranches horaires nécessairement travaillées par la machine k.

LB_k^1 : Nombre de couples fixés par probing LB_k^1 est obtenue en sommant les couples (k, tranche horaire s) fixés par probing (cf. définition 3) :

$$LB_k^1 = card\{s \in S | (k, s) \text{ est fixé par probing}\}$$

LB_k^2 : Durée des opérations Le calcul de LB_k^2 est basé sur l'amplitude minimale de la période globale d'utilisation de k. On exploite ici l'idée qu'une machine ne peut exécuter plus d'un job par instant. La machine k est alors nécessairement utilisée durant exactement D_k instants, $D_k = \sum_{i \in J} \rho_{ik}$ étant la somme des durées des opérations que k doit exécuter. D_k constitue ainsi une borne inférieure de l'amplitude de la période globale d'utilisation de k. LB_k^2 est alors définie comme suit :

$$LB_k^2 = \left\lceil \frac{D_k}{\pi} \right\rceil$$

où π est, rappelons-le, la durée de chaque tranche horaire $s \in S$.

LB_k^3 : Probing intensif LB_k^3 est calculée via une méthode de *probing* intensifiée, généralisant la technique de *probing* vue en 9.1. Dans cette idée, on cherche à ordonnancer, sur la machine k, un nombre incrémental $\sigma' \leq \sigma$ d'opérations fictives de durée π et de domaines d'exécution recouvrant l'horizon. Comme on l'a vu précédemment dans la technique de probing, un tel procédé permet de symboliser de manière indirecte des périodes d'inactivité de machine.

En considérant ainsi σ' opérations fictives, on obtient un problème $[JobShop_F]$ de production surcontraint, ayant pour objectif d'ordonnancer chaque job $i \in J$ et chaque opération fictive. $[JobShop_F]$ est un problème classique de job-shop dont nous testons la réalisabilité en appliquant les techniques décrites en 8.4.

Si $[JobShop_F]$ n'est pas réalisable, chaque opération fictive étant de durée π, cela implique que la machine k ne peut être laissée au repos durant au moins σ' tranches horaires. $LB_k^3 = (\sigma - \sigma')$ est alors une borne inférieure du nombre minimal de tranches horaires nécessairement travaillées sur k.

L'algorithme 7 décrit les étapes principales du calcul de LB_k^3.

Algorithme 7 Calcul de LB_k^3

$LB_k^3 \leftarrow 0$; $\sigma' \leftarrow 1$; irréalisable $\leftarrow$ faux
Répéter
  $O_F \leftarrow \sigma'$ opérations fictives de durée π et devant être exécutées sur k
  $[JobShop_F] \leftarrow$ problème de job-shop devant ordonnancer chaque job $i \in J$ et chaque opération de O_F
  Si $[JobShop_F]$ n'est pas réalisable **alors**
    irréalisable $\leftarrow$ vrai
  Sinon
    $\sigma' \leftarrow \sigma' + 1$
  Fin Si
Jusqu'à (irréalisable) $\vee$ $(\sigma' > \sigma)$
$LB_k^3 \leftarrow \sigma - \sigma'$

9.2.3 Expression de la coupe

Forts du calcul, pour chaque machine k, des trois valeurs LB_k^1, LB_k^2 et LB_k^3, nous pouvons exprimer les inégalités (9.2) qui invalident les solutions pour lesquelles le nombre de tranches horaires travaillées par machine est trop faible.

$$\sum_{e \in E | k \in A_e} \sum_{s \in T_e} x_{eks} \geq LB_k \quad k = 1, \ldots, m \tag{9.2}$$

où $LB_k = \max(LB_k^1, LB_k^2, LB_k^3)$

LB_k^3 domine clairement LB_k^1 et LB_k^2. Pour prendre cette domination en compte dans notre implémentation, nous calculons tout d'abord LB_k^1 et LB_k^2 puis initialisons LB_k^3 à $\max(LB_k^1, LB_k^2)$ dans l'algorithme 7.

9.3 Inégalités initiales complémentaires

Les deux ensembles de coupes initiales 9.1 et 9.2 décrits ci-dessus ont pour objectif commun de forcer le travail sur tout ou partie de sous-ensembles de couples (machine, tranche horaire). Elles imposent alors qu'un nombre minimal d'employés soit affecté à ces sous-ensembles. Ces inégalités n'interdisent cependant pas le recouvrement d'un même couple (machine, tranche horaire) par plusieurs employés. Il est pourtant simple de constater que, les coûts d'affectation d'employés étant tous positifs et chaque machine ne nécessitant qu'un employé pour être pilotée, il est sous-optimal d'attribuer plus d'un employé à une machine durant une tranche horaire donnée. Nous introduisons alors les coupes *complémentaires* 9.3 afin de s'assurer qu'au plus un employé ne soit affecté à chaque couple (machine, tranche horaire).

$$\sum_{e \in E | (k \in A_e) \wedge (s \in T_e)} x_{eks} \leq 1 \quad k = 1, \ldots, m \quad s = 0, \ldots, \delta - 1 \tag{9.3}$$

Clairement, toute solution affectant plus d'un employé à un même couple (machine, tranche horaire) étant sous-optimale pour le problème originel, les inégalités 9.3 peuvent a priori sembler de moindre utilité. Ce n'est en réalité pas le cas, ce groupe de contraintes permet en effet de mieux guider la recherche et sert efficacement l'élimination plus rapide de solutions sous-optimales. Diverses expérimentations numériques prouvent leur efficacité pour chacune de nos méthodes.

Chapitre 10

Décomposition et génération de coupes

Dans ce chapitre, nous définissons une technique de résolution analogue à la technique de décomposition et génération de coupes mise en place pour résoudre la première problématique à laquelle nous nous sommes intéressés durant cette thèse (cf. chapitre 4). Rappelons ici qu'une des motivations initiales pour l'étude de cette seconde problématique intégrant planification d'agents et ordonnancement de production était d'éprouver la technique de décomposition et génération de coupes, efficace pour nos premiers travaux, sur un problème plus complexe. On s'intéresse désormais à un problème de production *NP-complet* (*job-shop*), à la différence du problème de *flot maximal* que l'on résolvait en un temps polynomial auparavant.

Cette première méthode de résolution, exploitant la génération de coupes de réalisabilité, s'articule parfaitement avec les inégalités valides définies au chapitre précédent. Ces dernières constituent en effet un ensemble de coupes initiales valides compatibles avec notre démarche. Comme on le verra par la suite, la symbiose avec les inégalités valides est même renforcée par le fait que les coupes de réalisabilité générées en cours de résolution sont analogues aux inégalités initiales de *probing* (9.1) définies en 9.1.3.

Nous présentons tout d'abord le fonctionnement global de la méthode de décomposition et de génération de coupes appliquée à cette seconde problématique. Les deux sous-problèmes découlant naturellement de la décomposition intuitive du modèle complet sont alors définis à tour de rôle. Le schéma de génération de coupes est lui ensuite exposé, juste avant l'algorithme général de la méthode.

10.1 Processus global

En raison de sa structure en deux niveaux de décision, analogue à celle de la première problématique étudiée lors de cette thèse, il semble là encore naturel d'expérimenter une technique de décomposition pour résoudre le modèle $[PL]$ proposé en 7.3.5. La méthode exacte décrite ici reprend les fondements de la technique de coupes définie dans le chapitre 4. La décomposition de $[PL]$ est ici exploitée en deux sous-problèmes $[ETP]$ [8] et $[JobShop]$. Le problème maître $[ETP]$ définit tout d'abord une affectation complète des employés aux machines et aux tranches horaires (notons-la $\bar{x}$). Utilisant ces informations comme données, le sous-problème $[JobShop]$ vérifie ensuite qu'un plan d'ordonnancement global puisse être défini à partir des ressources en compétences définies par $\bar{x}$. Si $[JobShop]$ ne possède aucune solution réalisable, $\bar{x}$ n'est pas une solution réalisable de $[PL]$. Une coupe invalidant $\bar{x}$ est alors ajoutée à $[ETP]$. $[ETP]$ fournissant, à chaque itération, l'affectation réalisable de machines et de tranches horaires aux employés de moindre coût, la première affectation conduisant à une solution réalisable dans $[JobShop]$ est optimale ; le processus est alors arrêté. Si $[PL]$ est irréalisable, le processus itère jusqu'à ce que $[ETP]$ n'ait plus de solution réalisable.

10.2 Définition du problème maître (ETP)

$[ETP]$ est un problème de gestion d'emplois du temps. $[ETP]$ affecte au moindre coût des machines et des tranches horaires de travail aux employés, les contraintes couplantes de ressources liant les employés à l'organisation de la production étant relaxées. Une formalisation en termes de Programmes Linéaires en Nombres Entiers de $[ETP]$, basée sur le vecteur de variables x et sur les contraintes non couplantes en lien avec ces variables dans la formalisation $[PL]$, est alors :

$$
\begin{aligned}
[ETP]: \quad \min \Theta = & \sum_{e\in E}\sum_{k\in A_e}\sum_{s\in T_e} c_{eks}\cdot x_{eks} & & \\
& \sum_{k\notin A_e}\sum_{s=0}^{\sigma-1} x_{eks} = 0 & e=1,\dots,\mu & \\
& \sum_{k\in A_e}\sum_{s\notin T_e} x_{eks} = 0 & e=1,\dots,\mu & \\
& \sum_{k\in A_e}\left(x_{eks}+x_{ek(s+1)}+x_{ek(s+2)}\right)\leq 1 & e=1,\dots,\mu & s=0,\dots,\sigma-3 \\
& \textit{Coupes} & & \\
& x_{eks}\in\{0,1\} & e=1,\dots,\mu & k=1,\dots,m \\
& & s=0,\dots,\sigma-1 &
\end{aligned}
$$

8. *ETP* étant acronyme de Employee Timetabling Problem

Coupes désignant l'ensemble des coupes de réalisabilité ajoutées itérativement au modèle. Ces coupes invalident les solutions de $[ETP]$ ne permettant pas d'obtenir une solution complète réalisable pour $[PL]$.

Dans nos expérimentations, $[ETP]$ est résolu par un solveur de PLNE (Ilog Cplex 11.2).

10.3 Définition du sous-problème $(JobShop)$

Considérons une affectation $\bar{x}$ de machines et de tranches horaires aux employés comme une solution optimale de $[ETP]$.

Afin de juger de la réalisabilité de $\bar{x}$ pour $[PL]$, il convient de définir un sous-problème dont le but est de vérifier qu'un ordonnancement de production compatible avec le niveau en ressources humaines imposé par $\bar{x}$ existe.

Avant de définir un tel sous-problème, on s'intéresse tout d'abord à aggréger l'information contenue dans la solution $\bar{x}$. On définit pour ce faire un vecteur $\bar{z}$ indiquant la présence ou non, selon $\bar{x}$, d'un employé sur chaque couple (machine, tranche horaire). $\bar{z}$ est ainsi défini comme suit :

$$\bar{z}_k^s = \min(\sum_{e \in E} \bar{x}_{eks}, 1) \quad k = 1, \ldots, m, \quad s = 0, \ldots, \sigma - 1$$

Par définition : $\bar{z}_k^s = 1$ si un employé est affecté, selon $\bar{x}$, au couple (machine k, tranche horaire s), 0 sinon.

Proposition 9 *La machine k peut être utilisée durant la tranche horaire s si et seulement si $\bar{z}_k^s = 1$.*

Preuve.
Pour un couple (machine k, tranche horaire s) donné, $\bar{z}_k^s \neq 1$ si et seulement si aucun employé ne travaille sur k durant s. Chaque machine nécessitant un employé pour être utilisée, il apparaît alors clairement que pour tout couple (machine k, tranche horaire s), k ne pourra être utilisée que si $\bar{z}_k^s = 1$. ■

Vérifier la réalisabilité de $\bar{x}$ pour $[PL]$ revient à tester qu'un ordonnancement réalisable pour le problème de production de type job-shop, et respectant la distribution en machines utilisables $\bar{z}$ existe. Le sous-problème $[JobShop(\bar{z})]$ est alors introduit :

$[JobShop(\bar{z})]$: *Existe-t-il un ordonnancement vérifiant :*

$$
\begin{aligned}
&\sum_{t=0}^{d_{ik}-\rho_{ik}} t \cdot y_{ikt} + \rho_{ik} \leq C_{max} && i = 1,\dots,n \quad k = m_{im} \\
&\sum_{t=r_{ik}}^{d_{ik}-\rho_{ik}} y_{ikt} = 1 && i = 1,\dots,n \quad k = 1,\dots,m \\
&\sum_{t=0}^{r_{ik}} y_{ikt} + \sum_{t=d_{ik}-\rho_{ik}+1}^{C_{max}} y_{ikt} = 0 && i = 1,\dots,n \quad k = 1,\dots,m \\
&\sum_{u=r_{ik}+\rho_{ik}}^{t} y_{ilu} - \sum_{u=r_{ik}}^{t-\rho_{ik}} y_{iku} \leq 0 && i = 1,\dots,n \quad j = 1,\dots,m-1 \\
& && k = m_{ij} \quad l = m_{i(j+1)} \\
& && t = r_{ik}+\rho_{ik},\dots,d_{il}-\rho_{il} \qquad (10.1) \\
&\sum_{i=1}^{n} \sum_{u=\max(r_{ik},t-\rho_{ik}+1)}^{\min(d_{ik}-\rho_{ik},t)} y_{iku} \leq 1 && k = 1,\dots,m \quad t = 0,\dots,C_{max} \\
&\sum_{i=1}^{n} \sum_{t=\max(r_{ik},\bar{s}\cdot\pi-\rho_{ik})}^{\min(C_{max},(s+1)\cdot\pi)} y_{ikt} = 0 && k = 1,\dots,m \quad \bar{v} \in \bar{S}_k \qquad (10.2) \\
&y_{ikt} \in \{0,1\} && i = 1,\dots,n \quad k = 1,\dots,m \\
& && t = 0,\dots,C_{max}
\end{aligned}
$$

$\bar{S}_k = \{s \in S | \bar{z}_k^s = 0\}$ désignant l'ensemble des tranches horaires pour lesquelles la machine $k \in [\![1, m]\!]$ ne peut pas être utilisée selon $\bar{z}$.

$[JobShop(\bar{z})]$ repose essentiellement sur les contraintes impliquant les variables y de la formalisation $[PL]$. Seul le groupe de contraintes 10.2, tributaire de la distribution $\bar{z}$, est propre à ce nouveau modèle ; ces égalités assurent qu'aucun job n'est en cours d'exécution sur chaque couple (machine, tranche horaire) imposé à l'inactivité par $\bar{z}$.

$[JobShop(\bar{z})]$ est un problème de job-shop à périodes fixées d'inactivité (cf. chapitre 8). Dans nos expérimentations, nous exploitons la réécriture de $[JobShop(\bar{z})]$ en un problème classique de job-shop décrite en 8.3. Le problème résultant est ensuite résolu exactement en appliquant les techniques propres au problème de job-shop proposées en 8.4.

10.4 Schéma de génération de coupes

Si $[JobShop(\bar{z})]$ est irréalisable, cela prouve qu'il est impossible d'ordonnancer chaque job $i \in J$ à partir des ressources définies par $\bar{z}$. $\bar{z}$ étant directement défini par $\bar{x}$, cela signifie que $\bar{x}$ n'est pas une solution réalisable du problème global $[PL]$ et doit alors être éliminé de l'ensemble des solutions admissibles.

Trivialement, si $\bar{z}$ n'est pas réalisable, c'est parce qu'au moins une des machines en inactivité durant une certaine tranche horaire devrait être utilisée.

La coupe invalidant $\bar{x}$ qu'il convient d'ajouter à l'ensemble *Coupes* de $[ETP]$ s'écrit alors :

$$\sum_{e \in E} \sum_{k \in A_e} \sum_{s \in T_e} \bar{\beta}_k^s \cdot x_{eks} \geq 1 \tag{10.3}$$

où $\bar{\beta}_k^s = 1$ si $\bar{z}_k^s \neq 1$, 0 sinon.

Nous remarquons ici une forte similitude structurelle entre les coupes de réalisabilité 10.3 et les inégalités valides de *probing* (9.1) (décrites en 9.1.3) générées en pré-process.

10.5 Algorithme

L'algorithme 8 résume la méthode exacte de décomposition et génération de coupes décrite ci-dessus.

Algorithme 8 Méthode exacte de génération de coupes

$LB \leftarrow 0$
Répéter
  $\bar{x} \leftarrow$ solution optimale de $[ETP]$, de coût : $\Theta_{\bar{x}}$
  Si $\bar{x}$ est définie **alors**
    $LB \leftarrow \Theta_{\bar{x}}$
    $\forall (k,s) \in \{M \times S\} \quad \bar{z}_k^s \leftarrow \min(\sum_{e \in E} \bar{x}_{eks}, 1)$
    réalisable $\leftarrow$ résoudre$[JobShop(\bar{z})]$
    Si $\neg$ réalisable **alors**
      ajouter la coupe (10.3) à l'ensemble *Coupes* de $[ETP]$
    Fin Si
  Fin Si
Jusqu'à (réalisable) $\vee$ ($\bar{x}$ n'est pas définie)
Renvoyer LB

Chapitre 11

Branch and Cut

Nous proposons dans ce chapitre une approche exacte alternative à la technique de décomposition et génération de coupes définie au chapitre précédent. La méthode présentée ici repose sur une technique hybridant une approche arborescente de type Procédure de Séparation et Évaluation Séquentielle et la méthode de coupes vue précédemment. L'idée directrice de cette nouvelle approche consiste à s'appuyer sur les fondements de la technique de coupes (à savoir : la décomposition de $[PL]$ en les deux mêmes sous-problèmes et la génération de coupes de réalisabilité) dans un cadre où les temps de calcul induits par la recherche, à chaque itération de la méthode de coupes, d'une solution réalisable complète pour le problème de planification d'agents sont réduits.

Après avoir présenté les grandes lignes de cette approche hybride, nous définissons les caractéristiques de la méthode arborescente utilisée (le schéma de branchement, des règles d'élimination et d'implication ainsi que la fonction d'évaluation des nœuds). Nous formulons ensuite un test de consistance effectué en tout nœud que cette technique exacte juge intéressant d'explorer ; selon le résultat de ce test, une solution réalisable pour $[PL]$ est définie ou une coupe de réalisabilité est générée. L'algorithme général de la méthode est introduit en fin de ce chapitre.

11.1 Processus global

À l'instar de la méthode de décomposition et génération de coupes vue au chapitre précédent, nous exploitons ici la même décomposition du problème global $[PL]$ en les deux sous-problèmes $[ETP]$ et $[JobShop(\bar{z})]$. La technique décrite ici diffère par la manière de générer le vecteur $\bar{z}$, $\bar{z}$ définissant une distribution de machines utilisables par tranche horaire et servant d'entrées au sous-problème $[JobShop(\bar{z})]$. À l'inverse de la génération de coupes où $\bar{z}$ est défini exhaustivement à chaque itération à partir de la solution optimale de $[ETP]$, nous proposons ici de construire $\bar{z}$ itérativement via une méthode arborescente de type Procédure de Séparation et Évaluation Séquentielle.

Le schéma de séparation retenu est basé sur le travail ou non en un couple donné (machine, tranche horaire). Chaque nœud de l'arborescence constitue alors un plan d'occupation partiel des machines dont la réalisabilité doit être prouvée à la fois au niveau de la planification d'agents (via le sous-problème $[ETP]$) et au niveau de l'ordonnancement de production (via le sous-problème $[JobShop(\bar{z})]$).

Si l'un au moins de ces deux sous-problèmes prouvent que la distribution partielle en cours n'est pas réalisable, aucun nœud descendant du nœud courant ne conduira à une distribution complète réalisable ; ce dernier est alors éliminé.

En revanche, si la distribution partielle est a priori réalisable pour les deux sous-problèmes, un test de consistance ayant pour objet de tester la réalisabilité directe du nœud en cours est mis en place. Dans cette idée, on fixe au repos tous les couples (machine, tranche horaire) pour lesquels aucune décision n'a été prise jusqu'alors : on obtient alors une distribution complète de travail sur les couples (machine, tranche horaire). Les sous-problèmes $[ETP]$ et $[JobShop(\bar{z})]$ sont ensuite résolus optimalement. Si ces sous-problèmes sont tous les deux réalisables et que le coût engendré par cette distribution complète dans $[ETP]$ est inférieur à la meilleure borne supérieure connue, on dispose d'une solution réalisable améliorante pour $[PL]$. Sinon, la coupe de réalisabilité (10.3) (cf. 10.4) est ajoutée à l'ensemble *Coupes* de $[ETP]$ (cette coupe étant valide en tout nœud de l'aborescence) et les deux fils du nœud courant sont créés et ajoutés à la liste des nœuds devant être explorés.

Le processus itère jusqu'à ce que chacun des nœuds de l'arborescence ait été exploré ou éliminé. Notons que cette méthode arborescente constructive permet d'obtenir, en cours de résolution, des solutions réalisables (s'il en existe) du modèle complet $[PL]$.

11.2 Branchement

Le schéma de branchement de la méthode arborescente est un schéma binaire portant sur le travail d'un couple donné (machine k, tranche horaire s). D'un côté, on impose l'inactivité de la machine k durant s et de l'autre, on oblige la présence d'un employé sur k durant s.

Concrètement, on utilise un vecteur $\bar{z}$ (analogue au vecteur $\bar{z}$ de la technique de décomposition et génération de coupes vue au chapitre précédent). $\bar{z}$ a pour rôle d'indiquer la présence ou non d'un employé sur chaque couple (machine, tranche horaire). $\bar{z}$ est ainsi défini tel que :

$$\forall (k,s) \in \{K \times S\} \quad \bar{z}_{ks} = \begin{cases} -1 & \text{si le travail sur } (k,s) \text{ n'est pas contraint} \\ 0 & \text{si la machine } k \text{ doit être utilisée durant } s \\ 1 & \text{si la machine } k \text{ ne doit pas être utilisée durant } s \end{cases}$$

On explore tout d'abord la branche de l'arbre de recherche inhérente au nœud fils symbolisant l'absence de travail. Le nœud exprimant l'obligation de travail est ensuite étudié. Ce schéma de branchement a été retenu car interdire le travail sur un couple (machine, tranche horaire) réduit davantage l'espace des solutions réalisables. Il est alors a priori plus probable que le nœud en cours soit jugé irréalisable et donc éliminé.

11.3 Sélection de la variable de branchement

Conformément au schéma de branchement retenu, nous sélectionnons un couple (machine $\bar{k}$, tranche horaire $\bar{s}$) défini tel que :

- $\bar{k} \in M$ est la machine dont l'écart entre le nombre de tranches horaires pouvant ou devant être travaillées et la borne inférieure LB_k (cf. 9.2) du nombre minimal de tranches horaires devant être travaillées est maximal. Formellement, $\bar{k}$ est sélectionnée telle que :

$$\bar{k} = \text{argmax}_{k \in K | \exists s \in S, \bar{z}_k^s = -1} \left(\text{card} \left\{ (k,s) \in \{K \times S\} \, | \bar{z}_k^s \in \{-1, 1\} \right\} - LB_k \right)$$

- $\bar{s} \in S$ est la tranche horaire la plus proche de l'horizon de planification telle qu'aucune décision n'a encore été prise quant au travail, ou non, d'un employé sur la machine $\bar{k}$. On a formellement :

$$\bar{s} = \text{argmax} \left(s \in S | \bar{z}_{\bar{k}}^s = -1 \right)$$

La stratégie retenue au travers de cette sélection de variables est la stratégie s'étant avérée la plus performante expérimentalement.

La machine $\bar{k}$ est la machine pour laquelle un maximum de tranches horaires non fixées peuvent a priori être laissées au repos. Nous sélectionnons cette machine en priorité car, dans la mesure où la variable de branchement est tout d'abord fixée au repos ($\bar{z}_k^s = 0$), il est raisonnable de supposer qu'imposer le repos sur cette machine ne rendra pas le problème irréalisable. La méthode arborescente convergera alors plus rapidement vers une solution complète réalisable.

Pour sa part, la tranche horaire $\bar{s}$ est choisie de préférence en fin d'horizon car c'est a priori sur la fin de l'ordonnancement que le problème est le plus contraint. De ce fait, interdire le travail sur une tranche horaire proche de l'horizon impliquera plus de décisions directes sur d'autres couples (machine, tranche horaire) ; l'arbre de recherche s'en trouvant alors réduit.

11.4 Évaluation

L'évaluation d'un nœud de l'arbre de recherche a pour but de déterminer une borne de l'optimum de l'ensemble des solutions réalisables associé au nœud en question. Cette valeur constitue alors la valeur potentielle de la meilleure solution atteignable dans la branche en cours. Si cette borne est moins intéressante (selon la fonction-objectif du problème) que la meilleure solution trouvée jusqu'à présent, il est inutile d'effectuer la séparation de l'espace de solutions du nœud courant ; ce dernier est alors éliminé.

Dans nos expérimentations, la fonction d'évaluation est la relaxation d'un sous-problème $[ETP(\bar{z})]$ dérivant du sous-problème de gestion d'emplois du temps $[ETP]$ de la méthode de coupes. Seules les contraintes de ressources (11.1) liées aux décisions de branchement prises au cours de l'exploration de l'arbre de recherche sont ajoutées à $[ETP]$. Une formalisation en termes de Programme Linéaire en Nombres Entiers de $[ETP(\bar{z})]$ est alors :

$$
\begin{aligned}
[ETP(\bar{z})] : \min \quad & \Theta_{\bar{z}} = \sum_{e\in E}\sum_{k\in A_e}\sum_{s\in T_e} c_{eks}\cdot x_{eks} & & \\
& \sum_{k\notin A_e}\sum_{s=0}^{\sigma-1} x_{eks} = 0 & & e=1,\ldots,\mu \\
& \sum_{k\in A_e}\sum_{s\notin \mathcal{T}_e} x_{eks} = 0 & & e=1,\ldots,\mu \\
& \sum_{k\in A_e}\left(x_{eks}+x_{ek(s+1)}+x_{ek(s+2)}\right)\leq 1 & & e=1,\ldots,\mu \quad s=0,\ldots,\sigma-3 \\
& \sum_{e\in E} x_{eks} = \bar{z}_k^s & & \forall (k,s)\in\{\{K\times S\}\,|\bar{z}_k^s\neq -1\} \\
& \textit{Coupes} & & \\
& x_{eks}\in\{0,1\} & & e=1,\ldots,\mu \quad k=1,\ldots,m \\
& & & s=0,\ldots,\sigma-1
\end{aligned}
\tag{11.1}
$$

L'évaluation par la relaxation continue de $[ETP(\bar{z})]$ a un deuxième intérêt dans notre approche. Résoudre la relaxation continue de $[ETP(\bar{z})]$ permet en effet également de vérifier que la distribution $\bar{z}$ vérifie les contraintes relaxées continûment du problème de gestion d'emplois du temps ainsi que les coupes de réalisabilité ajoutées itérativement au modèle. Si ce n'est pas le cas, $\bar{z}$ ne permettra pas d'atteindre une solution complète réalisable ; le nœud courant est alors éliminé.

11.5 Règles d'élimination et d'implication

Afin de réduire l'espace de recherche, différentes règles d'élimination et d'implication ont été mises en place. Ces règles ont pour rôle premier d'analyser, en chaque nœud, les décisions héritées de ses ancêtres dans l'arbre de recherche. Forts de ces informations, il est quelquefois possible -pour garantir la réalisabilité du problème en cours dans la branche étudiée- de fixer certaines variables de décision. Dans d'autres cas, il peut même être prouvé que les décisions prises jusqu'alors contraignent chaque descendant du nœud courant à mener à une solution complète irréalisable ; dans de tels cas, le nœud courant est éliminé et ses descendants ne sont pas explorés.

11.5.1 Élimination - implication par probing

La première règle que nous définissons est basée sur la stratégie de probing (cf. 9.1). On cherche à vérifier que les décisions successives quant au travail ou non de certains couples (machine, tranche horaire) prises jusqu'alors n'ont pas réduit l'espace des solutions réalisables à l'ensemble vide.

Pour un nœud donné de l'arbre de recherche, on résout le même sous-problème $[Jobshop(\bar{z})]$ que la méthode de décomposition et génération de coupes présentée au chapitre précédent ; i.e. un problème de job-shop à périodes fixées d'inactivité où chaque période contrainte au repos correspond à un couple (machine k, tranche horaire s) imposé à l'absence de travail ($\bar{z}_k^s = 0$). Si $[JobShop(\bar{z})]$ ne parvient pas à trouver un ordonnancement réalisable, la solution partielle $\bar{z}$ ne peut pas mener à une solution complète réalisable. Le nœud courant est alors éliminé.

Par suite, si le nœud n'est pas éliminé, on applique exactement la stratégie de probing vue en 9.1 afin de déterminer des couples (machine, tranche horaire) fixés par probing. Pour cela, on s'intéresse au sous-problème $[JobShop(\bar{z})]$ défini ci-dessus. Chacun à leur tour, on interdit ensuite dans $[JobShop(\bar{z})]$ le travail sur les couples (machine, tranche horaire) non fixés. $[JobShop(\bar{z})]$ est alors résolu ; si aucun ordonnancement compatible n'existe, le couple (machine, tranche horaire) en cours est *fixé par probing* ; il sera nécessairement travaillé dans tout nœud de la branche courante de l'arbre de recherche.

11.5.2 Implication par calcul du nombre minimal de tranches horaires devant être travaillées

Une seconde règle d'implication portant sur le nombre minimal de tranches horaires devant être travaillées peut être mise en place.

Dans cette nouvelle idée, nous vérifions en chaque nouveau nœud que, pour toute machine $k \in M$, la borne inférieure LB_k du nombre minimal de tranches horaires

devant être travaillées sur k est bien respectée[9]. Si ce n'est pas le cas, le nœud courant est éliminé.

Si la limite du nombre de tranches horaires non travaillées est atteinte (mais non dépassée), nous forçons, pour chaque nœud descendant, le travail sur l'ensemble des tranches horaires non encore fixées sur k. Cette règle d'implication se formalise, pour k, comme suit :

$$(\text{card}\,\{s \in S | \bar{z}_k^s = 0\} = LB_k) \Rightarrow (\bar{z}_k^s = 1 \quad \forall s \in \{S | \bar{z}_k^s = -1\})$$

La mise en place algorithmique de cette seconde règle d'implication s'optimise. Cette étape de contrôle n'est opérante que lorsque le nombre de couples (machine, tranche horaire) pouvant être travaillées est inférieur à ceux du nœud père. Sans cela, les calculs sont identiques à ceux déjà effectués pour le nœud père. Concrètement, la règle n'est alors appliquée que pour les nœuds dont le schéma de branchement a directement imposé le repos sur un couple (machine, tranche horaire) donné.

11.6 Test de consistance

À tout nœud de la méthode arborescente correspond une distribution partielle $\bar{z}$ de travail, ou non, des couples (machine, tranche horaire). Si l'évaluation d'un tel nœud tend à prouver que sa distribution partielle $\bar{z}$ peut mener à une solution complète améliorante au problème de gestion d'emplois du temps, il est admis que ce dernier doit être séparé et ses fils explorés.

Ceci étant, une distribution complète peut aussi être facilement définie en tout nœud. Pour ce faire, il suffit d'imposer au repos tous les couples (machine, tranche horaire) non fixés. Formellement :

$$\bar{z}_k^s = 0 \quad \forall (k, s) \in \{K \times S | \bar{z}_k^s = -1\}$$

Les mêmes sous-problèmes $[ETP(\bar{z})]$ et $[Jobshop(\bar{z})]$, respectivement définis pour l'évaluation et les règles d'élimination-implication de chaque nœud, sont alors résolus optimalement. Si ces deux sous-problèmes prouvent que $\bar{z}$ est une distribution complète permettant de constituer une solution réalisable de $[PL]$ de coût inférieur à la meilleure solution connue jusqu'alors, on dispose d'une solution améliorante. Sinon, la coupe de réalisabilité (10.3) détaillée en (10.4) est ajoutée à l'ensemble *Coupes* au problème générique de gestion d'emplois du temps $[ETP]$. Cette coupe, valide en tout nœud de l'arbre de recherche, guidera alors la méthode à ne plus s'intéresser à des branches menant à des distributions similaires à la distribution complète irréalisable en cours.

9. LB_k est définie en 9.2

11.7 Algorithmique

Nous exposons dans cette section l'algorithmique impliquée dans le codage de la méthode arborescente. Deux structures informatiques de données utiles à notre code sont tout d'abord spécifiées. Nous détaillons ensuite l'algorithme de chaque étape de la technique arborescente.

11.7.1 Structure de données

Dans les algorithmes que nous avons développés, nous avons eu recours à deux structures informatiques de données particulières :

- P : une pile classique, respectant l'ordre First In Last Out.
- γ : un nœud de l'arborescence. Chaque nœud γ de l'arbre (informatique) de recherche possède 5 attributs :
  - $k \in M$: machine inhérente à la variable de branchement sélectionnée pour construire γ
  - $s \in S$: tranche horaire impliquée dans la variable de branchement sélectionnée du nœud γ
  - *exploré* $\in \{$*vrai, faux*$\}$: état d'exploration du nœud. Si la valeur est *vrai*, le nœud a déjà été exploré et ses deux fils ont été éventuellement créés. Si la valeur est *faux*, c'est la première fois que le nœud est étudié.
  - *valeur* $\in \{-1, 0, 1\}$: valeur fixée pour la variable de branchement liée à γ
  - *decisions* : liste des décisions (dûe à la sélection de la variable de branchement et aux implications de ce choix) prises pour le nœud γ. Ces décisions étant toutes, par construction, respectées par chaque descendant de γ.

11.7.2 Algorithme

Les algorithmes 9 à 14 synthétisent les étapes de calcul de la méthode arborescente exposée dans ce chapitre.

L'algorithme 9 est l'algorithme générique de la méthode, il décrit dans les grandes lignes les principes de la technique arborescente. Il a recours aux algorithmes 10 à 14 proposés ci-après :

- algorithme 10 : règles d'implication
- algorithme 11 : fonction d'évaluation
- algorithme 12 : schéma de branchement
- algorithme 13 : suppression d'un nœud
- algorithme 14 : teste si le nœud courant est une feuille

Algorithme 9 Algorithme de la méthode arborescente couplée à une génération de coupes

$UB \leftarrow +\infty$
$\forall k \in M \ \forall s \in S \ \bar{z}_k^s \leftarrow -1$
$P \leftarrow \{$Nœud (k=0, s=0, valeur=-1, exploré=faux, decisions=$\emptyset$)$\}$
Répéter
  $\gamma \leftarrow$ nœud au sommet de P
  Si $\neg$ γ.exploré **alors**
    γ.exploré $\leftarrow$ vrai
    $\bar{z}_{\gamma.k}^{\gamma.s} = \gamma.valeur$
    Si $[JobShop(\bar{z})]$ n'est pas irréalisable **alors**
      implications$(\gamma, \bar{z})$
      f $\leftarrow$ évaluer$(\gamma, \bar{z})$
      Si $(f \neq -1) \wedge (f < UB)$ **alors**
        $\forall (k,s) \in \{K \times S\} \quad (\bar{z}_k^s)' \leftarrow \bar{z}_k^s$
        $\forall (k,s) \in \{K \times S | \bar{z}_k^s = -1\} \quad \bar{z}_k^s \leftarrow 0$
        $\bar{x} \leftarrow$ solution optimale de $[ETP(\bar{z})]$, de coût : $\Theta_{\bar{z}}$
        Si ($\bar{x}$ est définie) $\wedge$ $(\Theta_{\bar{z}} < UB) \wedge$ ($[JobShop(\bar{z})]$ est réalisable) **alors**
          $UB \leftarrow \Theta_{\bar{z}}$
        Sinon
          ajouter la coupe (10.3) à l'ensemble *Coupes* de $[ETP]$
        Fin Si
        $\forall (k,s) \in \{K \times S\} \quad \bar{z}_k^s \leftarrow (\bar{z}_k^s)'$
        Si $\neg$ feuille$(\bar{z})$ **alors**
          brancher$(P, \bar{z})$
        Sinon
          supprimer$(P, \bar{z})$
        Fin Si
      Sinon
        supprimer$(P, \bar{z})$
      Fin Si
    Sinon
      supprimer$(P, \bar{z})$
    Fin Si
Jusqu'à $P = \emptyset$
Renvoyer UB

Algorithme 10 Règles d'implication : implications($\gamma, \bar{z}$)

Si $\gamma.valeur = 0$ **alors**
  Si card $\left\{s \in S | \bar{z}^s_{\gamma.k} = 0\right\} = LB_{\gamma.k}$ **alors**
    Pour tout $s \in S$ **Faire**
      Si $\bar{z}^s_{\gamma.k} = -1$ **alors**
        $\bar{z}^s_{\gamma.k} \leftarrow 1$
        $\gamma.decisions \leftarrow \gamma.decisions \cup \left\{\bar{z}^s_{\gamma.k} = 1\right\}$
      Fin Si
    Fin Pour
  Fin Si
  Pour tout $(k, s) \in \{K \times S\}$ **Faire**
    Si $\bar{z}^s_{\gamma.k} = -1$ **alors**
      $\bar{z}^s_k \leftarrow 0$
      Si $[JobShop(\bar{z})]$ est irréalisable **alors**
        $\bar{z}^s_k \leftarrow 1$
        $\gamma.decisions \leftarrow \gamma.decisions \cup \left\{\bar{z}^s_{\gamma.k} = 1\right\}$
      Sinon
        $\bar{z}^s_k \leftarrow -1$
      Fin Si
    Fin Si
  Fin Pour
Fin Si

Algorithme 11 Évaluation : évaluer($\gamma, \bar{z}$)

résultat ← -1
$\bar{x} \leftarrow$ solution optimale de la relaxation continue de $[ETP(\bar{z})]$, de coût : LP
Si $\bar{x}$ est définie **alors**
  résultat ← LP
Fin Si
Renvoyer résultat

Algorithme 12 Branchement : brancher($P, \bar{z}$)

$\bar{k} \leftarrow \text{argmax}_{k \in K | \exists s \in S, \bar{z}^s_k = -1} \left(\text{card}\left\{(k, s) \in \{K \times S\} \,|\, \bar{z}^s_k \in \{-1, 1\}\right\} - LB_k\right)$
$\bar{s} \leftarrow \text{argmax}\left(s \in S | \bar{z}^s_{\bar{k}} = -1\right)$
$P \leftarrow P \cup \text{Nœud}\left(\text{k=}\bar{k}, \text{ s=}\bar{s}, \text{ valeur=1, decisions=}\left\{\bar{z}^{\bar{s}}_{\bar{k}} = 1\right\}\right)$
$P \leftarrow P \cup \text{Nœud}\left(\text{k=}\bar{k}, \text{ s=}\bar{s}, \text{ valeur=0, decisions=}\left\{\bar{z}^{\bar{s}}_{\bar{k}} = 0\right\}\right)$

Algorithme 13 Supprimer : supprimer($P, \bar{z}$)

$\forall d \in \gamma.decisions \quad \bar{z}_{d.k}^{d.s} \leftarrow -1$
$P \leftarrow P \setminus \gamma$

Algorithme 14 Feuille : feuille($\bar{z}$)

Pour tout $(k, s) \in \{K \times S\}$ **Faire**
  Si $\bar{z}_k^s = -1$ **alors**
    Renvoyer faux
  Fin Si
Fin Pour
Renvoyer vrai

Chapitre 12

Expérimentations numériques

Les méthodes de résolution présentées dans cette seconde partie du manuscrit de la thèse ont été implémentées puis testées expérimentalement sur les instances proposées par Artigues et al. dans [AGRV09]. Nous présentons et discutons les résultats numériques de chacune de ces méthodes dans ce chapitre.

Dans un premier temps, nous détaillons le matériel informatique utilisé. Les caractéristiques de chaque instance de test sont exposées après cela. Les résultats des expérimentations numériques sont ensuite regroupés dans des tableaux illustratifs.

12.1 Matériel

Les techniques de résolution décrites dans cette partie ont été implémentées en Java et testées sur un PC (Intel Core 2 Duo CPU T9400, 2.53 GHz, 3.45 GB RAM) dont le système d'exploitation est MS Windows XP.

Le solveur de programmation linéaire (MIP) utilisé est ILOG CPLEX 11.2 ; les paramètres du solveur étant ceux par défaut.

12.2 Instances de test

Notre jeu de données est composé des 8 instances de [AGRV09] compatibles avec notre problématique. Les paramètres de ces instances sont décrits dans le tableau 12.1.

Les colonnes n et m du tableau 12.1 correspondent respectivement au nombre de jobs et au nombre de machines de l'instance. μ indique le nombre total d'employés disponibles. Parmi eux, on en considère $Extra$, à coût d'affectation très élevé, pouvant piloter n'importe quelle machine à n'importe quel moment. Ces $Extra$ employés sont une assurance de réalisabilité de chaque instance. Cela correspond en pratique à employer des intérimaires qualifiés durant des périodes de temps réduites. Chaque

Instance	n	m	μ	$Extra$	# maximal de compétences par employé	δ	π	C_{max}	Θ^*
ejs4	6	4	25	10	2	8	8	40	23
ejs9	6	4	25	10	2	8	8	49	24
ejs10	6	4	25	10	2	8	8	48	23
ejs8 $\times$ 8_1	8	8	40	20	4	8	8	44	78
ejs8 $\times$ 8_2	8	8	40	20	4	8	8	51	96
ejs8 $\times$ 8_3	8	8	40	20	4	8	8	44	83
ejs10 $\times$ 10_1	10	10	50	20	4	10	10	80	124
ejs10 $\times$ 10_3	10	10	50	20	4	10	10	84	95

TABLE 12.1 – Données

employé, en dehors des *Extra intérimaires*, maîtrisent au plus un certain nombre de compétences (colonne # maximale de compétences par employé).

La colonne δ représente le nombre de tranches horaires de l'horizon de planification des employés, π étant la durée de chacune d'elles. Les colonnes C_{max} et Θ^* indiquent enfin respectivement la borne supérieure autorisée pour la durée totale de l'ordonnancement et le coût en main d'œuvre de la solution optimale de l'instance.

12.3 Inégalités initiales

Nous nous intéressons tout d'abord à l'intérêt des inégalités initiales générables en préprocess (cf. chapitre 9). Nous comparons pour cela, dans le tableau 12.2, les résultats obtenus lors du calcul de la relaxation continue associée au modèle $[PL]$ selon que les coupes initiales sont, on ne sont pas, utilisées.

Dans le tableau 12.2, les colonnes Θ^* et LP indiquent respectivement le coût en main d'œuvre de la solution optimale et la valeur optimale de la relaxation continue de $[PL]$.

Dans la partie *Relaxation continue de [PL]*, la colonne *Tps* présente les temps de résolution de la relaxation continue dans le cas où les coupes initiales ne sont pas utilisées.

Dans l'autre partie du tableau 12.2 (*Relaxation continue de [PL] avec coupes initiales*), la colonne # *coupes* indique le nombre d'inégalités initiales générées en préprocess ; la colonne *Tps préprocess* exprimant le temps de calcul nécessaire pour les générer. Pour sa part, la colonne *Tps LP* montre le temps propre à la résolution de la relaxation continue du modèle $[PL]$ associé aux inégalités initiales. La colonne *Tps total* indique enfin le temps global nécessaire à nos calculs, soit le temps passé en préprocess additionné au temps de résolution du modèle continu (*Tps total* = *Tps préprocess* + *Tps LP*).

		Relaxation continue de $[PL]$			Relaxation continue de $[PL]$ avec coupes initiales				
Instance	Θ^*	LP	LP/Θ^*	Tps	LP	LP/Θ^*	#coupes	Tps préprocess	Tps LP	Tps total
ejs4	23	21.2	92.4%	0.4s	23.0	**100.0%**	43	0.0s	0.2s	0.2s
ejs9	24	20.9	87.3%	0.4s	22.7	**94.6%**	67	1.5s	0.3s	1.8s
ejs10	23	22.3	97.1%	0.4s	23.0	**100.0%**	50	0.4s	0.2s	0.6s
ejs8 $\times$ 8_1	78	46.9	60.1%	1.8s	67.2	**86.2%**	94	9.4s	1.4s	10.8s
ejs8 $\times$ 8_2	96	65.3	68.0%	2.4s	88.0	**91.6%**	103	7.9s	1.9s	9.8s
ejs8 $\times$ 8_3	83	45.7	55.1%	1.5s	76.2	**91.8%**	93	4.9s	1.2s	6.1s
ejs10 $\times$ 10_1	124	69.1	55.7%	12.4s	115.5	**93.1%**	158	18.0s	6.4s	24.4s
ejs10 $\times$ 10_3	95	65.7	68.4%	19.5s	86.9	**90.6%**	164	231.7s	11.1s	242.8s

TABLE 12.2 – Résultats de la relaxation continue

Nous remarquons, en analysant le tableau 12.2, que les coupes initiales ont un impact fort sur la relaxation continue associée au modèle $[PL]$. Lorsque ces dernières sont utilisées, le ratio LP/Θ^* est nettement amélioré pour l'ensemble des instances (de 20.5% en moyenne, et jusqu'à 37.4% pour $ejs10 \times 10_1$). Cette amélioration qualitative se fait cependant au prix d'un temps de calcul en moyenne 4.4 fois plus élévé ; 71.5% de ce temps étant passé à la génération des inégalités initiales.

12.4 Méthodes exactes

Nous présentons désormais les résultats des méthodes exactes décrites dans cette seconde partie du manuscrit. Nous reportons dans un premier temps les performances de trois techniques exactes : le solveur MIP appliqué à la formalisation $[PL]$ (MIP), la technique de décomposition et de génération de coupes ($Coupe$) et le Branch and Cut ($BranchAndCut$).

La limite de temps CPU imposée pour chaque méthode est de 5 minutes.

Un premier tableau 12.3 fait état des résultats de chaque méthode exacte lorsque les inégalités initiales présentées dans le chapitre 9 ne sont pas utilisées. Le tableau 12.4 synthétise les résultats de ces mêmes méthodes quand elles ont recours aux inégalités initiales ; les colonnes supplémentaires *Préprocess.# coupes* et *Préprocess.Tps* indiquant alors respectivement le nombre d'inégalités initiales créées et le temps nécessaire à leur génération.

Dans chacun de ces deux tableaux, les colonnes *Tps*, *#nœuds* et *# coupes* indiquent respectivement le temps de résolution (TL [10] si l'instance n'a pas été résolue optimalement), le nombre de nœuds de l'arbre de recherche (uniquement pour les méthodes MIP et $BranchAndCut$) et le nombre de coupes dédiées (10.3) ajoutées en cours de résolution (uniquement pour les méthodes $Coupe$ et $BranchAndCut$). La colonne Θ donne, pour sa part, la valeur de la solution trouvée par chaque méthode. En cas de succès, les trois méthodes renvoient l'optimum ; en cas d'échec (temps de résolution imparti dépassé), MIP et $Coupe$ retournent toutes deux une borne inférieure au contraire de $BranchAndCut$ qui renvoie une borne supérieure (ou -1 si aucune solution réalisable n'a été trouvée).

Nous montrons également dans le tableau 12.4 les meilleurs résultats d'Artigues et al. présentés dans [AGRV09]. Ces chiffres sont simplement évoqués à titre indicatif. Notre matériel informatique étant différent, une comparaison directe serait en effet dénuée de sens. Le matériel utilisé par Artigues et al. est le suivant : implémentation en C++ sur une architecture 64-bit AMD avec le système d'exploitation Linux ; solveur MIP utilisé : Ilog Cplex 9.1 ; solveurs PCC utilisés : Ilog Solver 6.1 et Ilog Scheduler 6.1).

10. TL étant acronyme de Time Limit

		MIP			Coupe			BranchAndCut			
Instance	Θ^*	Θ	Tps	# nœuds	Θ	Tps	# coupes	Θ	Tps	# nœuds	# coupes
ejs4	23	23	0.2s	0	3	TL	740	23	**0.0s**	35	16
ejs9	24	24	**7.3s**	427	3	TL	617	24	61.0s	711	313
ejs10	23	23	**0.5s**	0	3	TL	669	23	2.3s	61	25
ejs8 × 8_1	78	85	TL	1997	2	TL	396	78	**149.4s**	157	62
ejs8 × 8_2	96	-1	TL	761	2	TL	401	96	TL	453	122
ejs8 × 8_3	83	83	TL	2234	2	TL	400	83	**23.3s**	95	43
ejs10 × 10_1	124	-1	TL	95	2	TL	317	124	**91.8s**	167	76
ejs10 × 10_3	95	-1	TL	40	2	TL	287	102	TL	25	6

TABLE 12.3 – Résultats des méthodes exactes **sans** les coupes initiales

		Préprocess		MIP			Coupe			BranchAndCut				Artigues et al.	
Instance	Θ^*	# coupes	Tps	Θ	Tps	# nœuds	Θ	Tps	# coupes	Θ	Tps	# nœuds	# coupes	Θ	Tps
ejs4	23	43	0.0s	23	0.2s	0	23	**0.0s**	0	23	**0.0s**	3	0	23	*0.8s*
ejs9	24	67	1.5s	24	2.1s	9	24	**1.0s**	34	24	9.5s	71	29	24	*50.4s*
ejs10	23	50	0.4s	23	0.2s	0	23	**0.0s**	0	23	0.3s	13	2	23	*2.6s*
ejs8 × 8_1	78	94	9.4s	78	151.9s	2377	76	TL	268	78	**70.3s**	49	12	78	*81.2s*
ejs8 × 8_2	96	103	7.9s	96	**26.8s**	330	95	TL	340	96	104.8s	167	35	96	*103s*
ejs8 × 8_3	83	93	4.9s	83	146.5s	2039	83	**7.2s**	65	83	12.8s	33	9	83	*29s*
ejs10 × 10_1	124	158	18.0s	134	TL	232	124	173.8s	247	124	**22.2s**	23	4	124	*926s*
ejs10 × 10_3	95	164	231.7s	-1	TL	81	89	TL	373	-1	TL	11	1	150	*TL (1800s)*

TABLE 12.4 – Résultats des méthodes exactes **avec** les coupes initiales

Nous constatons tout d'abord, par l'analyse du tableau 12.3, que la méthode de décomposition et de génération de coupes (*Coupe*) ne résout aucune instance optimalement. Les bornes inférieures qu'elle affiche sont même d'une qualité moyenne (ratio moyen Θ/Θ^* égal à 6.2%). Cette méthode s'avère donc peu efficace pour cette seconde problématique. Une explication plausible à ceci réside dans la trop forte dégénérescence du problème maître [*ETP*]. Le sous-problème de gestion d'emplois du temps présente en effet la particularité d'avoir de nombreuses solutions différentes de même coût. La technique, visant à minimiser les coûts salariaux, sélectionne ainsi en priorité les nombreuses solutions de très faible coût où peu d'employés travaillent. *Coupe* peine alors à converger vers des solutions réalisables.

Nous observons ensuite que la méthode arborescente hybride (*BranchAndCut*) est la méthode exacte la plus robuste. Elle résout et prouve l'optimalité de 6 des 8 instances. À titre comparatif, le solveur MIP (*MIP*) ne trouve, pour sa part, la solution optimale que des 3 instances de plus petite taille. Cette méthode hybride, basée sur une structure de recherche arborescente et utilisant les concepts de la technique de décomposition et génération de coupes, est donc la méthode la plus appropriée à cette seconde problématique.

L'analyse du tableau 12.4 permet d'affirmer en premier lieu que les inégalités initiales améliorent de manière très significative les performances de chacune des 3 méthodes exactes. Le temps de calcul nécessaire à la génération de ces dernières est de plus très inférieur aux temps de résolution des méthodes. L'utilisation des inégalités initiales apporte donc une information conséquente au prix d'un moindre temps de calcul.

Nous remarquons après cela que la méthode hybride arborescente (*BranchAndCut*) s'avère à nouveau la plus robuste. Associée aux inégalités initiales, elle résout ici 7 des 8 instances, contrairement à la technique de coupes (*Coupe*) et au solveur MIP (*MIP*) qui n'en résolvent respectivement que 5 et 6. Le nombre de nœuds explorés dans l'arbre de recherche de *BranchAndCut* s'avère aussi, pour chaque instance de plus grande taille ($ejs8 \times 8_1 \rightarrow ejs10 \times 10_3$), nettement inférieur à celui du solveur MIP. Enfin, en termes de temps de calcul, *BranchAndCut* est également la méthode la plus efficace en moyenne.

Sur ces premiers résultats expérimentaux, effectués sur des instances connues de la littérature, nous observons donc que les différentes investigations menées au cours de la thèse pour cette deuxième problématique ont permis de mettre en place une méthode arborescente exacte efficace. Les expérimentations tendent en effet à démontrer que la technique mise en place domine les résultats obtenus avec l'un des meilleurs solveurs de programmation linéaire actuels. Les performances annoncées semblent de plus, a priori, sérieusement concurrencer les techniques dédiées existantes déjà décrites dans la littérature. Nous notons enfin l'intérêt tout particulier des inégalités initiales générées en pré-process de toute méthode de résolution. La

comparaison entre les tableaux 12.3 et 12.4 prouve de manière certaine leur importance pour chacune des méthodes décrites dans ce manuscrit.

Conclusion

Dans la seconde partie du manuscrit de cette thèse, nous nous sommes concentrés sur l'étude d'une seconde problématique couplant planification d'agents et ordonnancement de production. L'intérêt instigateur de ce travail était d'affirmer, ou infirmer, la performance de l'approche par décomposition et génération de coupes (si efficace pour la première problématique) pour une problématique intégrée plus complexe et connue de la littérature. Nous nous intéressons en effet ici à un problème étudié par Artigues et al. dans [AGRV09] dont le sous-problème d'ordonnancement de production est *NP-complet*.

Après avoir décrit puis proposé une formalisation en Programme Linéaire en Nombres Entiers de la problématique, nous avons défini et étudié un problème particulier de job-shop : le job-shop à périodes fixées d'inactivité. Nous avons vu que ce problème, central dans nos techniques de résolution, pouvait être réécrit en un problème classique de job-shop. La littérature disposant de nombreuses références (et donc de nombreuses approches de résolution efficaces) pour le problème classique de job-shop, une telle réécriture constitue une aubaine quant à la résolution du problème de job-shop à périodes fixées d'inactivité

Nous avons après cela défini des techniques de génération de coupes initiales ; ces inégalités pouvant servir à toute méthode de résolution. L'intérêt probant de ces coupes initiales a été démontré au travers d'expérimentations numériques (menées sur des instances de la littérature) révélatrices.

Deux techniques de résolution ont enfin été présentées. La première exploite directement les fondements de la technique de décomposition et de génération de coupes vue dans la première partie de la thèse, à savoir : une décomposition du problème global en deux sous-problèmes puis la résolution alternée de chacun des sous-problèmes avec, le cas échéant, une génération de coupes de réalisabilité. Devant des résultats expérimentaux décevants, nous avons dû conclure à une certaine inefficacité de l'approche. Une justification à ce relatif échec réside dans la propriété des coupes de réalisabilité. Elles n'ont pas, à l'inverse des coupes de réalisabilité de la technique dédiée à la première problématique, la caractéristique de réduire de manière importante le polyèdre des solutions du problème maître. Dès lors, ce même problème maître étant fortement dégénéré, la méthode peine à converger. Notre approche présentant néanmoins des concepts théoriques a priori intéressants, nous avons mis en place une approche exacte alternative, basée en partie sur la

technnique de décomposition et de génération de coupes. Cette seconde méthode est une approche arborescente de type Procédure de Séparation et Évaluation Séquentielle. En chaque nœud l'arbre de recherche est sélectionnée une variable de branchement servant à imposer le travail, ou le repos, à un couple donné (machine, tranche horaire). Une distribution partielle de machines utilisables par tranche horaire est ainsi définie en tout nœud. La réalisabilité a priori d'une telle distribution est testée sur les deux mêmes sous-problèmes de gestion d'emplois du temps et d'ordonnancement de production que la technique de décomposition et de génération de coupes. Si ces deux sous-problèmes sont réalisables, on impose momentanément le repos à chaque couple (machine, tranche horaire) non fixé jusqu'alors ; la distribution devient ainsi une distribution complète. Sa réalisabilité est testée pour chacun des deux sous-problèmes. Si la résolution d'aucun de ces deux sous-problèmes n'échoue, nous disposons d'une solution complète réalisable ; sinon une coupe de réalisabilité, identique à celle de la technique de décomposition et de génération de coupes, est générée et ajoutée au problème de gestion d'emplois du temps. Des règles d'élimination et d'implication sont également appliquées en chaque nœud afin d'assurer une meilleure convergence de la méthode. Expérimentalement, cette approche arborescente s'est montrée particulièrement performante, en comparaison avec la technique de décomposition et génération de coupes mais également avec l'un des meilleurs solveurs MIP commerciaux actuels (Ilog Solver 11.2).

Des nombreux prolongements possibles à ces travaux demeurent. Nous pouvons par exemple citer l'éventualité d'améliorer la technique de décomposition et de génération de coupes. Lors de la thèse, des tentatives ont été menées en ce sens. Une technique alternative a en effet été mise en place. Dans cette approche, nous tentons de *casser* la dégénerescence du problème maître de gestion d'emplois du temps. Nous recherchons pour cela l'affectation d'employés aux machines et aux tranches horaires, réalisable pour le problème de gestion d'emplois du temps, de moindre coût. Nous fixons ensuite ce coût et modifions la fonction-objectif du problème maître. Les solutions (à coût fixé) maximisant le nombre de couples (machine, tranche horaire) travaillés sont alors recherchées. Nous supposons alors que de telles solutions, proposant des effectifs en main d'œuvre plus importants, sont a priori les plus compatibles au problème d'ordonnancement de production. Dès lors que plus aucune solution au problème maître n'existe, nous relaxons la contrainte de coût et cherchons les solutions de coût plus élevé, puis itérons. Les résultats préliminiaires de cette approche alternative se sont malheureusement montrés peu probants ; nous avons alors préféré ne pas la présenter exhaustivement dans ce mémoire.

Un second prolongement à nos travaux consisterait à expérimenter nos méthodes de résolution sur des instances de taille plus conséquente, ou sur des instances de jobshop connues de la littérature (instances de la *OR Library* par exemple) auxquelles nous ajouterions des employés à disponibilité et niveau de qualification variés. Des régles différentes de résolution des sous-problèmes des méthodes exactes devraient alors probablement être mises en place.

Conclusion générale

Durant cette thèse de doctorat, deux problématiques de Recherche Opérationnelle ont été étudiées. Elles traitent toutes les deux du monde manufacturier et de ces difficultés à résoudre conjointement les deux problèmes organisationnels que sont la gestion des ressources humaines et la planification de la production.

Dans la première partie du mémoire, un cas particulier de problèmes intégrant ces deux facteurs décisionnels de l'entreprise a été étudié. Deux formalisations ont été proposées, chacune permettant de prendre implicitement en compte de nombreuses contraintes législatives, contractuelles, ...

Deux bornes inférieures par décomposition lagrangienne ont été décrites. Leurs résultats, dominés en qualité et en temps de calcul par la relaxation continue obtenue via un solveur de programmation linéaire, n'ont malheureusement pas été à la hauteur de nos espérances.

Deux techniques exactes de décomposition et génération de coupes ont ensuite été détaillées. La première, basée sur une décomposition de Benders, s'est avérée peu compétitive. La seconde, reposant sur une décomposition et une génération de coupes dédiées, s'est en revanche montrée particulièrement efficace.

Deux variantes de cette dernière technique exacte ont été implémentées. Une première variante résout de manière optimale, à chaque itération, un problème maître de *sac à dos multi-choix multidimensionnel* (*NP-difficile*) alors que la seconde trouve rapidement une solution réalisable en appliquant l'heuristique de *Feasibility Pump*. Convergeant également vers l'optimum, la seconde variante présente de plus l'avantage de fournir, en cours de processus, des solutions réalisables de qualité pour le problème global.

Une technique de génération d'inégalités valides, basée sur le *raisonnement énergétique*, a également été conçue pour cette problématique. Nous avons prouvé l'utilité de cette méthode, applicable en pré-process de toute technique de résolution, au travers de résultats expérimentaux éloquents.

Les travaux réalisés autour de cette première problématique ont fait l'objet d'un article [GLPR10a] et de plusieurs présentations en conférence [BGPR07, GLPR08a, GLPR08b, GLPR10b].

Les techniques de décomposition et génération de coupes s'étant avérées si efficaces pour la première problématique, nous avons entrepris d'expérimenter cette même approche sur une nouvelle problématique -plus générique et déjà traitée dans

la littérature- intégrant elle aussi planification d'agents et ordonnancement de production. Ces travaux font l'objet de la seconde partie de ce manuscrit.

Dans cette nouvelle problématique couplante, nous considérons un problème d'ordonnancement de production *NP-complet* : la version décisionnelle d'un problème de *job-shop*.

Forts des résultats encourageants obtenus par la génération d'inégalités valides lors de la première étude, nous avons implémenté, là aussi, une technique de génération de coupes valides applicable en pré-process de toute méthode de résolution. Une nouvelle fois, les résultats numériques expérimentaux ont confirmé que de telles inégalités initiales étaient un atout indéniable pour s'attaquer à cette classe de problèmes.

Une technique de décomposition et génération de coupes dédiées a ensuite été détaillée et implémentée. Ses résultats décevants nous ont malheureusement menés à conclure à un échec. Malgré une strucure décomposable analogue à celle de la première problématique, la technique de coupes n'est pas efficace. On peut avancer comme explication une trop forte dénaturation des programmes maîtres de la méthode. Les informations contenues ici par ces derniers ne permettent pas un contrôle poussé de l'ordonnancement de la production. Les coupes dédiées obtenues en retour de solutions irréalisables pour l'outil de production sont en effet relativement *pauvres* ; le nombre de solutions qu'elles invalident est faible. Cumulé à une forte dégénérescence des programmes maîtres, la méthode s'apparente alors quasiment à une méthode énumérative basique.

Non contents de ces résultats, nous avons mis en place une seconde méthode exacte. Cette dernière a pour prétention d'appliquer les concepts avantageux de la méthode de coupes, dans un contexte régulé par une construction réfléchie de solutions, dont la réalisabilité est à tester à la fois au niveau de la planification des agents et à celui de l'ordonnancement de la production. Nous avons, pour ce faire, recours à une méthode arborescente de type Procédure de Séparation et Évaluation Séquentielle. La réalisabilité des solutions construites à partir des décisions prises en chaque nœud de l'arbre de recherche est vérifiée pour les deux mêmes sous-problèmes que ceux de la technique de coupes. En cas d'irréalisabilité, les coupes dédiées définies pour la première méthode de résolution sont ajoutées au modèle global. Cette approche hybride s'est avérée compétitive vis à vis de l'un des meilleurs solveurs commerciaux actuels de programmation linéaire. Nous avons donc réussi à trouver un compromis a priori satisfaisant, pour cette seconde étude, entre l'information que l'on peut retirer par la décomposition intuitive de ce genre de problématiques, et la manière de générer les solutions dont nos méthodes se nourrissent pour générer des coupes dédiées aussi informatives que possible.

Les résultats obtenus sur cette seconde problématique ont été présentés en conférence [GLPR09, GLPR10c].

En conclusion générale de ce manuscrit, il nous paraît désormais clair que les techniques de décomposition et génération de coupes ont un rôle important à jouer dans la résolution de problèmes intégrant planification d'agents et ordonnancement

de production. Deux cas particuliers de cette classe de problèmes, à décomposition naturelle intuitive, ont en effet été résolus efficacemment par ce genre de méthodes durant la thèse dont les travaux sont présentés dans ce manuscrit.

Cet avis doit toutefois être précisé car les méthodes de coupes mises en place pour les deux problématiques étudiées ici n'ont pas montré des degrés de performance équivalents. Si dans la première étude, la technique directe de décomposition et génération de coupes a été une franche réussite, elle a été un échec pour la seconde. Pour cette deuxième étude, il a alors fallu définir (via une hybridation avec une méthode arborescente) un cadre plus strict d'utilisation des concepts de la méthode originelle de décomposition et génération de coupes.

La raison majeure, à notre avis, de l'échec de la technique directe de coupes dans la seconde étude réside dans la nature des inégalités de réalisabilité générées. Dans nos premiers travaux, les coupes de réalisabilité mettent en exergue des zones temporelles précises de conflit où l'outil de production a besoin de plus d'employés. Dans la seconde étude, ce genre d'informations n'est pas recherché ; seule la réponse oui/non quant à la réalisabilité du sous-problème de production est déterminée. Dès lors, en cas d'échec au niveau de l'ordonnancement de production, une quantité limitée d'information est retournée au problème maître. La coupe de réalisabilité générée ne force pas le travail sur des zones de conflit identifiées ; elle se contente d'indiquer que l'outil de production a besoin d'employés sur au moins une des zones chômées dans la solution courante. Très peu de solutions du problème maître sont alors retirées du polyèdre des solutions admissibles, la méthode peine donc à converger. Nous pensons par conséquent que, pour être efficaces, les techniques de coupes ne doivent pas se contenter de tester la réalisabilité de toute solution partielle ; elles doivent analyser et comprendre les raisons d'un éventuel échec. Plus cette analyse est fine, plus le polyèdre des solutions admissibles est réduit par l'inégalité de réalisabilité générée.

Nous pouvons aussi affirmer que pour chaque approche de décomposition et de génération de coupes, se réserver du temps pour générer des inégalités initiales diversifiées en pré-process nous semble être toujours profitable.

Bibliographie

[AB97] Hesham K. Alfares and James E. Bailey. Integrated project task and manpower scheduling. *IIIE Transactions*, 29 :711–717, 1997.

[AB07] Tobias Achterberg and Timo Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1) :77–86, March 2007.

[AGR07] Christian Artigues, Michel Gendreau, and Louis-Marie Rousseau. A flexible model and a hybrid exact method for integrated employee timetabling and production scheduling. *CORS / Optimization Days 2006 Joint Conference*, 3867 :67–84, 2007.

[AGRV09] Christian Artigues, Michel Gendreau, Louis-Marie Rousseau, and Adrien Vergnaud. Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound. *Computers & Operations Research*, 36 :2330–2340, 2009.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows - Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[BAL95] James Bailey, Hesham K. Alfares, and Win Yuan Lin. Optimization and heuristic models to integrate project task and manpower scheduling. *Computers and Industrial Engineering*, 29 :473–476, 1995.

[Ben62] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4 :238–252, 1962.

[BFL05] Livio Bertacco, Matteo Fischetti, and Andrea Lodi. A feasibility pump heuristic for general mixed-integer problems. Technical Report OR/05/5, DEIS - Università di Bologna, Italy, May 2005.

[BFS08] Philippe Baptiste, Marta Flamini, and Francis Sourd. Lagrangian bounds for just-in-time job-shop scheduling. *Computers & Operations Research*, 35(3) :906–915, 2008.

[BGAB83] Lawrence Bodin, Bruce L. Golden, Arjang A. Assad, and Michael O. Ball. Routing and scheduling of vehicles and crews - the state of the art. *Computers & Opertions Research*, 10 :63–212, 1983.

[BGPR07] Nadjib Brahimi, Olivier Guyon, Éric Pinson, and David Rivreau. Couplage planification-ordonnancement : une approche par décomposition et génération de coupes. In *FRANCORO V / 8° congrès de la société française de recherche opérationnelle et d'aide à la décision (ROADeF)*, Grenoble France, 02 2007.

[BLPN99] Philippe Baptiste, Claude Le Pape, and Wim Nuijten. Satisfiability tests and time-bound adjustments for cumulative scheduling problems. *Annals of Operations Research*, 92 :305–333, 1999.

[BM90] D.J. Bradley and J. B. Martin. Continuous personnel scheduling algorithms : a literature review. *Journal of the Society for Health Systems*, 2 :8–23, 1990.

[BN04] Philippe Baptiste and Emmanuel Néron. *Modèles et algorithmes en ordonnancement*. Ellipses, 2004.

[BPR03] Philippe Baptiste, Laurent Péridy, and David Rivreau. A branch and bound to minimize the number of late jobs on a single machine with release date constraints. *European Journal of Operational Research*, 144(1) :1–11, 2003.

[CC88] Jacques Carlier and Philippe Chrétienne. *Problèmes d'ordonnancement : modélisation, complexité, algorithmes*. Masson, 1988.

[CP89] Jacques Carlier and Éric Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35 :164–176, 1989.

[CP94] Jacques Carlier and Éric Pinson. Adjustement of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78(2) :146–161, 1994.

[CP95] Jacques Carlier and Laurent Péridy. Global adjustemens of heads and tails for the jobshop scheduling problem. In *Euro XIV*, 1995.

[CPPR04] Jacques Carlier, Laurent Péridy, Éric Pinson, and David Rivreau. *Handbook of scheduling : algorithms, models and performance analysis*, chapter 4- Elimination rules for job-shop scheduling problem : overview and extensions. CRC Press, 2004.

[CSSD01] Jean-François Cordeau, Goran Stojkovic, François Soumis, and Jacques Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35 :375–388, 2001.

[DDI+94] Guy Desaulniers, Jacques Desrosiers, Irina Ioachim, Marius M. Solomon, and François Soumis. A unified framework for deterministic time

constrained vehicle routing and crew scheduling problems. *Les Cahiers du GERAD*, 46, 1994.

[Det07] Boris Detienne. *Planification et odronnancement : méthodes de décomposition et génération de coupes*. PhD thesis, Université de Technologie de Compiègne, 2007.

[DM94] Richard L. Daniels and Joseph B. Mazzola. Flow shop scheduling with resource flexibility. *Operations Research*, 42 :504–522, 1994.

[DPPR09] Boris Detienne, Laurent Péridy, Éric Pinson, and David Rivreau. Cut generation for an employee timetabling problem. *European Journal of Operational Research*, 197 :1178–1184, 2009.

[EJKS04] Andreas T. Ernst, Houyuan Jiang, Mohan Krishnamoorthy, and David Sier. Staff scheduling and rostering : A review of applications, methods and models. *European Journal of Operational Research*, 153 :3 – 27, 2004.

[FF62] Lester Randolph Ford and Delbert Ray Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, 1962.

[FGL05] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104 :91–104, 2005.

[FH96] Christian Friberg and Knut Haase. An exact algorithm for the vehicle and crew scheduling problem. In *Computer-Aided Transit Scheduling. Volume 471 of Lecture Notes in Economics and Mathematical Systems*, 1996.

[FHW03] Richard Freling, Dennis Huisman, and Albert P. M. Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6 :63–85, 2003.

[Geo74] Arthur M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematicam Programming Study*, 2 :82–114, 1974.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and intractability : a guide to the theory of NP-Completness*. W.H. Freeman, 1979.

[GLPR08a] Olivier Guyon, Pierre Lemaire, Éric Pinson, and David Rivreau. Couplage planification/ordonnancement : une approche par décomposition et génération de coupes. In *7ième conférence internationale de modélisation et simulation MOSIM'08*, volume 2 of *Editions Tec&Doc*, pages 1376–1385, Paris France, 2008. Lavoisier.

[GLPR08b] Olivier Guyon, Pierre Lemaire, Éric Pinson, and David Rivreau. Planification d'agents et ordonnancement de production : règles d'élimination et heuristique. In *9° congrès de la société française de recherche opérationnelle et d'aide à la décision (ROADeF)*, Clermont-Ferrand France, 02 2008.

[GLPR09] Olivier Guyon, Pierre Lemaire, Éric Pinson, and David Rivreau. Résolution d'un problème de Job-Shop intégrant des contraintes de Ressources Humaines. In *10° congrès de la société française de recherche opérationnelle et d'aide à la décision (ROADeF)*, Nancy France, 02 2009.

[GLPR10a] Olivier Guyon, Pierre Lemaire, Éric Pinson, and David Rivreau. Cut generation for an integrated employee timetabling and production scheduling problem. *European Journal of Operational Research*, 201(2) :557–567, 2010.

[GLPR10b] Olivier Guyon, Pierre Lemaire, Éric Pinson, and David Rivreau. Near optimal and optimal solutions for an integrated employee timetabling and production scheduling problem. In Dolgui Alexandre Bakhtadze, Natalia, editor, *Information Control Problems in Manufacturing 13th IFAC Symposium on Information Control Problems in Manufacturing*, volume 13, pages 10.3182/20090603–3–RU–2001.0190, Moscou Russie, Fédération De, 02 2010.

[GLPR10c] Olivier Guyon, Pierre Lemaire, Éric Pinson, and David Rivreau. Solving an integrated Job-Shop problem with human resource constraints. In *PMS 2010, 12th International Workshop on Project Management and Scheduling*, Tours France, 04 2010.

[Hoo05] John N Hooker. A hybrid method for planning and scheduling. *Constraints*, 10 :385–401, 2005.

[Hoo07] John N Hooker. Planning and scheduling by logic-based benders decomposition. *Operations Research*, 55 :588–602, 2007.

[HWC74] Michael Held, Philip Wolfe, and Harlan P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6 :62–88, 1974.

[KJN+02] Diego Klabjan, Ellis J Johnson, George L Nemhauser, Éric Gelman, and Srini Ramaswamy. Airline crew scheduling with time windows and plane-count constraints. *Transportation Science*, 36 :337–348, 2002.

[LEE92] Pierre Lopez, Jacques Erschler, and Patrick Esquirol. Ordonnancement de tâches sous contraintes : une approche énergétique. *Automatique, Productique, Informatique Industrielle*, 26 :453–481, 1992.

[Leu04] Joseph Y-T. Leung. *Handbook of scheduling : algorithms, models and performance analysis*. CRC Press, Boca Raton, FL, USA, 2004.

[LRKB77] Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and Peter Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1 :343–362, 1977.

[MCS05] Anne Merciera, Jean-François Cordeau, and François Soumis. A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32 :1451–1476, 2005.

[MS03] Amnon Meisels and Andrea Schaerf. Modelling and solving employee timetabling problems. *Annals of Mathematics and Artificial Intelligence*, 39 :41–59, 2003.

[MT87] Silvano Martello and Paolo Toth. Algorithms for knapsack problems. *Annals of Discrete Mathematics*, 31 :70–79, 1987.

[MT90] Silvano Martello and Paolo Toth. *Knapsack problems ; algorithms and computer implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[P96] Laurent Péridy. *Le problème de job-shop : arbitrages et ajustements*. PhD thesis, Université de Technologie de Compiègne, 1996.

[Pin04] Michael Pinedo. *Scheduling : theory, algorithms and systems*. Prentice Hall, second edition, 2004.

[PPR97] Éric Pinson, Laurent Péridy, and David Rivreau. Elimination rules for disjunctive scheduling problems. In *Third Workshop on Models and Algorithms for Planning and Scheduling Problems*, Cambridge, April 1997.

[PR05] Laurent Péridy and David Rivreau. Local adjustments : a general algorithm. *European Journal of Operational Research*, 164 :24–38, 2005.

[Riv99] David Rivreau. *Problèmes d'ordonnancement disjonctifs : règles d'élimination et bornes inférieures*. PhD thesis, Université de Technologie de Compiègne, 1999.

[RK76] Alexander Hendrik George Rinnooy Kan. *Machine scheduling problems : classification, complexity and computation*. Kluwer Academic Publishers, 1976.

[Sav94] Martin W. P. Savelsbergh. Preprocessing and probing techniques for

mixed integer programming problems. *INFORMS Journal on Computing*, 6(4) :445–454, 1994.

[Sch99] Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13 :87–127, 1999.

[SPR05] François Soumis, Gilles Pesant, and Louis-Marie Rousseau. *Gestion de production et ressources humaines*, chapter 4, Gestion des horaires et affectation du personnel. Presses Internationales Polytechnique, 2005.

Résumé

Cette thèse consiste en l'étude de deux problèmes de la Recherche Opérationnelle, appliqués au monde manufacturier. Les problématiques étudiées traitent de l'intégration, dans le processus décisionnel industriel, de deux facteurs-clés : la planification des ressources humaines et l'ordonnancement de la production.

La première problématique porte sur un cas particulier de ce genre de problématiques. Deux bornes inférieures obtenues par décomposition lagrangienne et deux méthodes de résolution exacte par décomposition et génération de coupes sont présentées. Si la première approche relève d'une technique bien connue de la littérature (décomposition de Benders), la seconde se veut plus spécifique ; cette dernière présentant de plus l'avantage d'être déclinable en une variante heuristique. Une technique de génération de coupes *énergétiques* valides, applicable en préprocess de toute méthode de résolution, est également proposée.

La seconde partie traite d'un autre cas particulier, déjà évoqué dans la littérature, de la problématique générale. Ces travaux prolongent ceux effectués lors de la première étude dans le sens où le problème traité est intrinsèquement plus complexe et le but avoué est d'expérimenter les techniques de décomposition et génération de coupes, a priori efficaces, sur une autre problématique. Une technique de génération d'inégalités valides, applicable elle aussi en préprocess de toute méthode de résolution, est tout d'abord mise en place. Deux méthodes de résolution exacte sont ensuite développées. La première est analogue à la technique spécifique de décomposition décrite auparavant. La seconde, plus novatrice, exploite la décomposition intuitive de la problématique et la génération de coupes dédiées dans un cadre où les solutions à valider sont construites via une approche arborescente de type Procédure de Séparation et Évaluation Séquentielle.

Mots-clés:

Planification de personnel
Sac à dos multi-choix multidimensionnel
Relaxation lagrangienne
Décomposition et génération de coupes
Raisonnement énergétique
Probing
Ordonnancement de production
Job-shop
Décomposition de Benders
Branch and Cut
Feasibility Pump